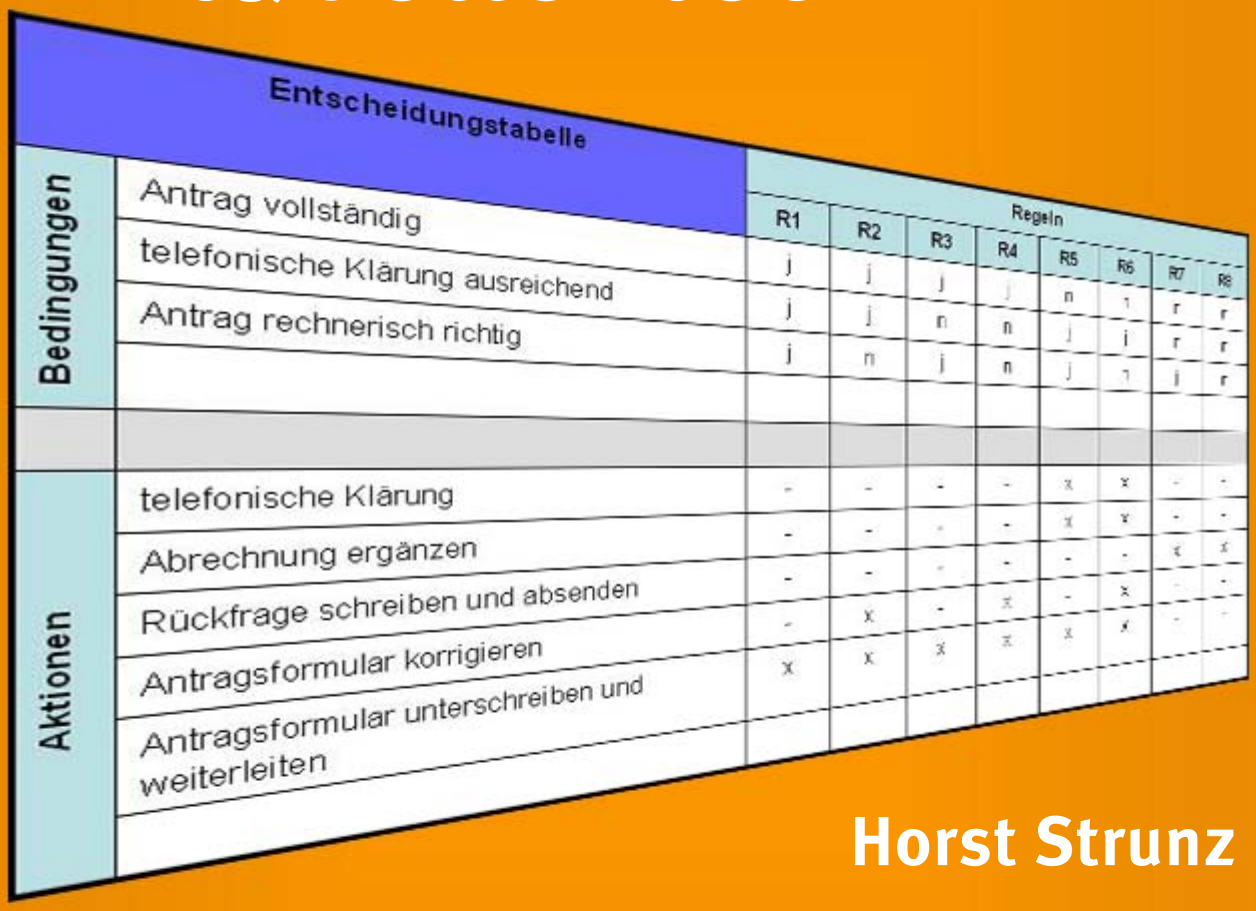
Entscheidungs-tabellentechnik



Horst Strunz

Grundlagen und Anwendungsmöglichkeiten bei der Gestaltung rechnergestützter Informationssysteme

`

European Master in Project Management

Strunz, Entscheidungstabellentechnik

European Master in Project Management

Elektronische Version herausgegeben von

Peter Reusch, Laura Esmeralda Garcia Moreno und dem EuroMPM Team

Entscheidungstabellentechnik

Grundlagen und Anwendungsmöglichkeiten bei der Gestaltung rechnergestützter Informationssysteme

von Horst Strunz

European Master in Project Management

Erstveröffentlichung 1977
ISBN 3-446-12382-2
Alle Verlagsrechte liegen beim Autor

Vorwort des Herausgebers

Das historische Buch von Horst Strunz wurde uns vom Autor überlassen, um es auch in Zukunft weiter in Lehre und Forschung, an Hochschulen und in der Praxis einsetzen zu können.

Im Rahmen des European Master in Project Management wurde das Buch von Laura Esmeralda Garcia Moreno eingescannt und redaktionell überarbeitet.

Parallel dazu wurde das Buch 2009 von Julia Gdaniec ins Englische übersetzt. Die Übersetzung wurde von Muhammad Khushnood redigiert.

Weitere Übersetzungen des Buches sind angestrebt.

Unser Entschluss, dieses historische Buch wieder verfügbar zu machen und sogar zu übersetzen ist wohl begründet:

- die Entscheidungstabellentechnik ist heute nach wie vor eine der wichtigsten Methoden der Systemanalyse und Systemspezifikation und wird auch durch neuere Entwicklungswerkzeuge wirksam unterstützt (ARIS®, ILOG-JRules®, LF-ET®, PROLOGA®, ...)

- das Buch von Horst Strunz ist ein Grundlagenwerk zur Entscheidungstabellentechnik, geschrieben vor mehr als dreißig Jahren und immer noch aktuell, weil sich an den Grundkonzepten und der Logik nichts geändert hat

- die Entscheidungstabellentechnik ist ein zentrales Werkzeug des Projektmanagement und ist in modernen Ausbildungskonzepten für Projektmanager unverzichtbar

- Entscheidungstabellentechnik spielt eine zentrale Rolle sowohl in der Lehre als auch in der Forschung im European Master in Project Management (EuroMPM) und wird an den Partnerhochschulen insbesondere in Dortmund, Bilbao, und Riga angewandt. Im Kontext des EuroMPM wurde auch die englische Übersetzung dieses Buches von Horst Strunz erstellt

- Entscheidungstabellen sind eingebunden in Objekt-Rollen-Modelle und Fact-Based-Modeling (Fact Based Modeling Foundation gegründet von Nijssen und Reusch).

Wir hoffen, dass auf diesem Weg das historische Buch wieder viele neue Freunde findet.

Wir danken Horst Strunz für die Zusammenarbeit – auch im Rahmen unserer Workshops in Dortmund.

Peter J. A. Reusch

Fachhochschule Dortmund

Vorwort des Verfassers

Über 40 Jahre beschäftigen wir uns bereits mit Methoden und Verfahren des Software Engineering und haben ein wichtiges Ziel noch immer nicht erreicht: Methoden, Verfahren und Tools bereitzustellen, die sicherstellen, dass die mit ihrer Hilfe entwickelte Software fehlerfrei ist.

Einer der Gründe dafür ist, dass wir bei der IT-Systementwicklung mit mehreren Sprachen zur Beschreibung der Geschäftsprozesse arbeiten: der zukünftige Anwender des Systems erläutert seine Anforderungen in einer Mischung aus natürlicher und Fach-Sprache, der Systemanalytiker erfasst im Rahmen des Requirements Engineering die aus seiner Sicht relevanten Sachverhalte in einer Notation, die auch vom Programmierer verstanden wird und dieser überführt die Vorgaben unter Nutzung einer Programmiersprache dann endlich in eine Form, die von Softwarewerkzeugen maschinell in lauffähige Software umgewandelt werden kann. Wir haben es dabei mit einem Prozess einer in mehreren Stufen erfolgenden Formalisierung verbunden mit einer zunehmenden Abstraktion zu tun. Durch den Übersetzungsprozess am Übergang von einer in die andere Sprache und durch die Verallgemeinerung von Sachverhalten können unbemerkt von den Beteiligten ganz wesentliche Details der Problemstellung untergehen. Die Fehler, die sich dadurch in die IT-Systeme einschleichen, können gravierende Auswirkungen haben.

Als ich 1969 als frischgebackener Assistent am BIFOA der Universität Köln im Projekt „Die Wirksamkeit von Programmiersprachen“ von einem amerikanischen Berater und PL/1-Spezialisten, der sich im Rahmen eines Beratungsprojektes in Deutschland aufhielt, das erste Mal eher beiläufig eine Entscheidungstabelle erläutert bekam, traf es mich wie ein Blitz. Ich hatte während meines Studiums meinen Lebensunterhalt als Werkstudent in der DV-Abteilung der Bayerwerke in Dormagen verdient und konnte mich noch sehr gut daran erinnern, dass ich wegen Programmstillstands eines meiner ersten Programme für die IBM 1401 vom Operator nachts aus dem Bett geholt wurde. In meinem Programm wurden 3 Banddateien verarbeitet und ich hatte nicht alle der 7 möglichen unterschiedlichen Dateiende-Konstellationen im Programm abgehandelt. Mein Ruf als hoffnungsvoller Jungprogrammierer war bereits erheblich angeknackst, als ich dann nach einem erneuten Programmstillstand endlich für alle logisch möglichen Fälle eine Programmroutine eingebaut hatte. Wäre mir damals die Entscheidungstabelle bereits vertraut gewesen, wäre mir einiger Frust erspart geblieben.

Die Entscheidungstabellentechnik hat in den 70er-Jahren vielleicht deshalb eine rasche Verbreitung gefunden, weil die zur Illustration der Methodik verwendeten Entscheidungstabellen-Beispiele bei vielen Lesern einen ähnlichen Aha-Effekt ausgelöst haben wie bei mir. In den vielen Diskussionen mit den Teilnehmern meiner über 200 durchgeführten Seminare und den Rückmeldungen aus der Anwendungspraxis wurden aber auch methodische Lücken deutlich, die ich durch fallweise Weiterentwicklungen schließen und auch in die Arbeit der Decision Tables Task Group der CODASYL einbringen konnte. Erst die Vorbereitung meiner Dissertation brachte dann aber den Durchbruch, nämlich die Erkenntnis, dass Entscheidungstabellen als Strukturs prache eingestuft werden müssen und die Weiterentwicklung einer Anwendungsmethodik erheblich davon profitieren könnte, wenn sie auf der Basis der Erkenntnisse der Semiotik in den Dimensionen Syntax, Semantik und Pragmatik untersucht würden.

Die vorliegende Arbeit ist der unveränderte Nachdruck einer aus meiner Dissertation von 1975 abgeleiteten Veröffentlichung aus dem Jahr 1977, die zur besseren Lesbarkeit von einigen vertiefenden und zur Verständlichkeit nicht unbedingt notwendigen mathematischen Ausführungen befreit worden war.

Der Teil I zur Geschichte der Entscheidungstabellentechnik stellt dar, wie die Grundlagen der Entscheidungstabellentechnik von den Pionieren in der Praxis entwickelt wurden. Meinen Ausführungen zu Entscheidungstabellen als Struktursprache im Teil II hätte ich auch heute – nach 34 Jahren - nichts mehr hinzuzufügen. Eine vollständige Definition der theoretischen Grundlagen der Entscheidungstabellentechnik im Rahmen einer Dissertation kann aber nicht gleichzeitig dem Anspruch genügen, den Leser in didaktisch ansprechender Weise und mit vielen anschaulichen Beispielen schrittweise in eine anspruchsvolle Materie einzuführen. Ich kann deshalb nur hoffen, dass das Interesse des Lesers an der Entscheidungstabellentechnik bereits vorher geweckt wurde und er – beispielsweise in der akademischen Ausbildung mit dem Ziel eines IT-Berufes oder als Praktiker vor einer Methodenentscheidung im Rahmen eines größeren IT-Projektes - eine Motivation mitbringt, dieser Methode auf den Grund zu gehen. Der Teil III befasst sich mit der Integration der Entscheidungstabellentechnik in die Praxis der IT-Systementwicklung. Wenn dabei auf konkrete Systementwicklungsmethoden und Eigenschaften von Programmiersprachen eingegangen wird, dann handelt es sich natürlich um Aussagen zum State-of-the-art Mitte der 70er-Jahre.

Nichts spricht aber gegen die Integration der Entscheidungstabellentechnik in die heute verbreiteten Systementwicklungsmethoden und Programmiersprachen.
Da die heutige Praxis der IT-Systementwicklung noch immer nicht sicherstellen kann, dass wir uns bei der Inbetriebnahme von IT-Anwendungen auf die Fehlerfreiheit der automatisierten Prozesse verlassen können, ist es gerechtfertigt, jeder brauchbaren Methode zur Qualitätsverbesserung von IT-Systemen Beachtung zu schenken. Es ist meine feste Überzeugung, dass die heutige IT-Welt um einige Probleme entlastet würde, wenn Entscheidungstabellen als ergänzende strukturelle Sprachelemente selbstverständlicher Bestandteil jeder Fachsprache wäre, die bei der Modellierung von Geschäftsprozessen eingesetzt wird. Sie versetzt den Systemanalytiker in die Lage, selbst aus rudimentären Angaben der Geschäftsprozess-Anwender Fälle zu konstruieren, deren praktische Relevanz im Gespräch mit den Praktikern verifiziert werden können. Die Entscheidungstabellentechnik unterstützt den Systemanalytiker also dabei, implizites Wissen des Anwenders zu Tage zu fördern. Auf diese Weise entsteht trotz lückenhaftem Domänen-Wissen auf Seiten des Systemanalytikers und möglicher fachlicher Kommunikationsbarrieren ein produktiver Dialog zwischen Geschäftsprozess-Anwender und Systemanalytiker, der in vertretbarer Zeit zu vollständigen und logisch konsistenten Spezifikationen der Ablauflogik von Geschäftsprozessen führt.

Eine Entscheidungstabelle kann von jedem nach einigen Minuten Einweisung richtig interpretiert werden. Leider hat die Historie gezeigt, dass dadurch auch die Erwartung geweckt wurde, dass man nach kurzer Einweisung in die Methodik auch in der Lage sei, komplexe Systeme mit Hilfe von Entscheidungstabellen zu spezifizieren. Auf die darauf folgende Frustration ist vielleicht zurückzuführen, dass das Interesse an der Entscheidungstabellentechnik in den 90-er Jahren wieder abgeebbt ist. Wer heute als IT-Systementwickler arbeitet, ist aber besser vorgebildet und akzeptiert, dass eine Systementwicklungsmethode in der Anwendung anspruchsvoll sein darf, wenn dadurch erhebliche Qualitätssteigerungen eines IT-Systems über seinen gesamten Lebenszyklus erreicht werden können.. Deshalb sind die Chancen einer Revitalisierung der Entscheidungstabellentechnik deutlich gestiegen und ich freue mich über jede Initiative, den Entscheidungstabellen als Struktursprache in Wissenschaft und IT-Praxis wieder einen höheren Stellenwert einzuräumen.

Rösrath, im Juli 2009

Horst Strunz

Inhaltsverzeichnis

Teil I: Geschichte der Entscheidungstabellentechnik 1

Teil II: Theoretische Grundlagen der Entscheidungstabellentechnik 11

1. Theorie der Eintreffer-Entscheidungstabelle 13
1.1 Syntax der Entscheidungstabelle 14
1.1.1 Bildungsregeln 14
1.1.1.1 Elementare syntaktische Einheiten 15
1.1.1.1.1 Bedingungen und Bedingungsanzeiger 15
1.1.1.1.2 Aktionen und Aktionsanzeiger 16
1.1.1.2 Höhere syntaktische Einheiten 17
1.1.1.2.1Basisfolge der Bedingungen 17
1.1.1.2.2 Basisfolge der Aktionen 17
1.1.1.2.3 Bedingungsanzeigerkombinationen 17
1.1.1.2.4 Aktionsanzeigerkombinationen 18
1.1.1.2.5 Entscheidungsregeln 19
1.1.1.2.6 Entscheidungstabellen 19
1.1.1.2.7 Entscheidungstabellenverbund 24
1.1.1.3 Bildung höherer syntaktischer Einheiten 27
1.1.1.3.1 Bildung einer Basisfolge von Bedingungen 27
1.1.1.3.2 Bildung einer Basisfolge von Aktionen 27
1.1.1.3.3 Bildung von Bedingungsanzeigerkombinationen 27
1.1.1.3.4 Bildung von Aktionsanzeigerkombinationen 31
1.1.1.3.5 Bildung von Entscheidungsregeln 32
1.1.1.3.6 Bildung von Entscheidungstabellen 33
1.1.1.3.7 Bildung des Entscheidungstabellenverbunds 35
1.1.1.4 Syntaktische Klassifikation von Entscheidungstabellen 37
1.1.1.4.1 Begrenzte, erweiterte und gemischte Entscheidungstabellen 37
1.1.1.4.2 Vollständige und unvollständige Entscheidungstabellen 39
1.1.1.4.3 Entscheidungstabellen in kanonischer Normalform 40
1.1.2 Transformationsregeln 42
1.1.2.1 Transformationen syntaktischer Einheiten in syntaktische Einheiten gleicher Stufe 42
1.1.2.1.1. Transformationen elementarer syntaktischer Einheiten 42
1.1.2.1.1.1 Bedingungstransformation 42
1.1.2.1.1.2 Aktionstransformation 43
1.1.2.1.2 Transformationen syntaktischer Einheiten höherer Stufe 43
1.1.2.1.2.1 Transformationen von Entscheidungsregeln 44
1.1.2.1.2.1.1 Elementare Regeltransformationen 44
1.1.2.1.2.1.2 Verfahren zur Regelminimierung 47
1.1.2.1.2.2 Transformationen von Entscheidungstabellen 52
1.1.2.1.2.2.1 Auflösung erweiterter und gemischter Entscheidungs-tabellen in begrenzte Entscheidungstabellen 53
1.1.2.1.2.2.2 Umwandlung von Entscheidungstabellen in ihre kanonische Normalform 55
1.1.2.2 Transformationen syntaktischer Einheiten in syntaktische Einheiten anderer Stufe 57
1.1.2.2.1 Zergliederung von Entscheidungstabellen 58
1.1.2.2.2 Kombination von Entscheidungstabellen 64

1.1.3 Syntaxprüfungen ... 72
1.1.3.1 Eindeutigkeitsprüfung ... 73
1.1.3.2 Vollständigkeitsprüfung ... 76
1.2 Semantik der Entscheidungstabelle ... 80
1.2.1 Abbildungsfunktion der Entscheidungstabelle ... 81
1.2.1.1 Abbildung der Aufbaustruktur des Entscheidungsfeldes ... 81
1.2.1.1.1 Abbildung der Bedingungen ... 81
1.2.1.1.2 Abbildung der Aktionen ... 82
1.2.1.1.3 Abbildung der Entscheidungsregeln ... 83
1.2.1.2 Abbildung der Ablaufstruktur des Entscheidungsfeldes ... 83
1.2.2 Verifikation der Entscheidungstabelle ... 85
1.2.2.1 Analyse semantischer Redundanzen und Mehrdeutigkeiten ... 86
1.2.2.1.1 Analyse der Basisfolge der Bedingungen ... 86
1.2.2.1.2 Analyse der Basisfolge der Aktionen ... 87
1.2.2.1.3 Analyse der Entscheidungsregeln ... 88
1.2.2.2. Analyse semantischer Unvollständigkeit ... 92
1.2.2.2.1 Ermittlung fehlender Bedingungen und Aktionen ... 93
1.2.2.2.2 Vollständigkeit bei der Abbildung von Bedingungszuständen und Aktionsausprägungen ... 96
1.2.2.2.3 Vollständigkeit bei der Abbildung der Bedingungskonstellationen ... 98
1.2.2.2.4 Vollständigkeit bei der Abbildung der Aktionsfolgen ... 106
1.2.3 Minimierung der Entscheidungstabelle ... 106
1.2.3.1 Minimierung von Bedingungen und Aktionen ... 107
1.2.3.2 Minimierung von Bedingungsanzeigern ... 110
1.2.3.3 Minimierung von Entscheidungsregeln ... 111
1.2.3.4 Minimierung des Entscheidungstabellenverbunds ... 111
1.2.4 Erstellung der Entscheidungstabelle ... 117
1.2.4.1 Das klassische Verfahren zur Erstellung von Entscheidungsstabellen ... 118
1.2.4.2 Tabellenerstellung durch fortschreitende Regelentwicklung ... 119
1.2.4.3 Erstellung des Entscheidungstabellenverbundes ... 130
1.3 Pragmatik der Entscheidungstabelle ... 133

2. Ansatz zu einer Theorie der Mehrtreffer-Entscheidungstabelle ... 134
2.1 Syntax der Mehrtreffer - Entscheidungstabelle ... 136
2.1.1 Bildungsregeln ... 136
2.1.1.1 Format der Mehrtreffer-Entscheidungstabelle ... 136
2.1.1.2 Bildung der Mehrtreffer-Entscheidungstabelle ... 138
2.1.1.3 Syntaktische Klassifikation von Mehrtreffer-Entscheidungstabellen ... 138
2.1.2 Transformationsregeln ... 140
2.1.2.1 Regeltransformationen ... 140
2.1.2.1.1 Elementare Regeltransformationen ... 140
2.1.2.1.2 Verfahren zur kombinierten Regel – und Anzeigerminimierung ... 149
2.1.2.1.2.1 Das Karnaugh -Diagramm ... 149
2.1.2.1.2.2 Das Quine-McCluskey-Verfahren und seine Modifikation ... 152
2.1.2.2 Formattransformationen ... 161
2.1.2.3 Transformationen zwischen Eintreffer-Entscheidungstabelle und Mehrtreffer-Entscheidungstabelle ... 164
2.1.3 Syntaxprüfungen ... 169

2.2 Semantik der Mehrtreffer –Entscheidungstabelle 171
2.2.1 Abbildungsfunktionen der Mehrtreffer-Entscheidungstabelle 171
2.2.2 Verifikation der Mehrtreffer-Entscheidungstabelle 171
2.2.2.1 Vollständigkeit bei der Abbildung der Bedingungskonstellationen 172
2.2.2.2 Vollständigkeit bei der Abbildung der Aktionsfolgen 172
2.2.2.3 Widerspruchsfreiheit der Aktionsfolgen 173
2.2.3 Minimierung der Mehrtreffer-Entscheidungstabelle 174
2.2.4 Erstellung der Mehrtreffer-Entscheidungstabelle 176
2.2.4.1 Erstellung der Mehrtreffer-Entscheidungstabelle 176
2.2.4.2 Bedingungsorientierte Regelentwicklung 177
2.2.4.3 Freie Regeleintragung 178
2.3 Pragmatik der Mehrtreffer-Entscheidungstabelle im Vergleich zur Eintreffer-Entscheidungstabelle 178
2.3.1 Erstellungs- und Verifikationsaufwand 178
2.3.2 Kommunikationseigenschaften 181
2.3.3 Konsequenzen für den praktischen Einsatz 182

Teil III: Anwendungsmöglichkeiten der Entscheidungstabellentechnik bei der Gestaltung rechnergestützter Informationssysteme 183

3. Der Systemcharakter rechnergestützter Informationssysteme und die Phasen deren Gestaltung 185
3.1 Elemente eines rechnergestützten Informationssystems 185
3.2 Ein Phasenmodell der Systemgestaltung 186

4. Entscheidungstabellen als Hilfsmittel der Systemplanung 191
4.1 Einsatzmöglichkeiten der Entscheidungstabellentechnik bei der Systemanalyse 191
4.1.1 Entscheidungstabellen in der Phase der Projektinitialisierung 191
4.1.2 Istaufnahme mit Entscheidungstabellen 192
4.2 Einsatzmöglichkeiten der Entscheidungstabellentechnik beim Systementwurf 193
4.2.1 Entscheidungstabelleneinsatz im Rahmen organisatorischer Entwurfsarbeiten 193
4.2.2 Systemtechnischer Entwurf mit Entscheidungstabellen 195
4.3 Grundlagen einer rechnergestützten Entwicklung von Entscheidungstabellen 198
4.3.1 Teilaufgaben der Entwicklung von Entscheidungstabellen 199
4.3.2 Anwendung graphentheoretischer Verfahren zur rechnergestützten Entwicklung von Entscheidungstabellen 201
4.3.2.1 Einige grundlegende Definitionen der Graphentheorie 201
4.3.2.2 Entscheidungstabelle und Entscheidungstabellenverbund als Graph 207
4.3.2.3 Graphentheoretische Lösung des Zergliederungs- und Modularisierungsproblems 213
4.3.3 Rechnergestützte Erstellung und Verifikation von Einzeltabellen 222

5. Entscheidungstabellen als Hilfsmittel der Systemimplementierung 226
5.1 Entscheidungstabellen als Hilfsmittel der systemtechnischen Implementierung 226
5.1.1 Teilaufgaben der systemtechnischen Implementierung von Informationsverarbeitungsprozessen 226
5.1.2 Methodische Programmierung.unter Einsatz von Entscheidungsstabellen 227
5.1.2.1 Modulare Programmierung 227
5.1.2.1.1 Definition der Modularen Programmierung 228
5.1.2.1.2 Gestaltungsprinzipien der Modularen Programmierung 229

5.1.2.1.3 Entscheidungstabellen im Konzept der Modularen Programmierung . 232
5.1.2.2 Strukturierte Programmierung ... 234
5.1.2.2.1 Merkmale der Strukturierten Programmierung ... 234
5.1.2.2.1.1 Elementare Ablaufstrukturen ... 234
5.1.2.2.1.2 "GO TO" -lose Programmierung ... 237
5.1.2.2.1.3 Konzept der Abstraktionsebenen ... 239
5.1.2.2.1.4 Top –down -Programmierung ... 240
5.1.2.2.2 Entscheidungstabellen im Konzept der Strukturierten Programmierung 241
5.1.2.3 Gedanken zur Integration von Modularer Programmierung, Strukturierter Programmierung und Entscheidungstabellentechnik ... 248
5.1.3 Der Einfluss der Entscheidungstabellen-Anwendung auf die Programm-Übersetzung ... 251
5.1.3.1 Konzepte der Behandlung von Entscheidungstabellen in einem Programm ... 252
5.1.3.2 Algorithmentypen zur Übersetzung von Entscheidungstabellen ... 254
5.1.3.2.1 Verfahren zur manuellen Codierung von Entscheidungstabellen ... 255
5.1.3.2.1.1 Codierung "Regel-für-Regel" ... 255
5.1.3.2.1.2 Entscheidungsbaum-Verfahren ... 258
5.1.3.2.1.3 Veinott - Methode ... 261
5.1.3.2.1.4 Optimierung der Aktionen -Codierung ... 263
5.1.3.2.2 Algorithmen für Entscheidungstabellenübersetzer ... 264
5.1.3.2.2.1 Entscheidungsbaum -Algorithmen ... 264
5.1.3.2.2.2 Maskenverfahren ... 271
5.1.4 Der Beitrag der Entscheidungstabellentechnik zur Lösung des Problems der Verifikation von Programmen ... 274
5.1.4.1 Stufen des Verifikationsprozesses ... 275
5.1.4.2 Verifikationsmethoden und –techniken ... 276
5.1.4.3 Entscheidungstabellentechnik als Verifikationstechnik ... 279
5.1.5 Programmoptimierung bei Einsatz von Entscheidungstabellen ... 280
5.1.6 Fallbeispiel zum Einsatz eines Entscheidungstabellen -Vorübersetzers ... 283
5.2 Entscheidungstabellen als Hilfsmittel der organisatorischen Implementierung ... 290
5.2.1 Teilaufgaben der organisatorischen Implementierung ... 291
5.2.2 Einsatzmöglichkeiten der Entscheidungstabellentechnik ... 291

6. Auswirkungen der Anwendung von Entscheidungstabellen bei der Systemanwendung ... 293
6.1 Zuverlässigkeit der Programme ... 293
6.2 Wartungsfreundlichkeit der Programme ... 295

7. Schluss ... 297
8. Verzeichnis der Abbildungen ... 298
9. Literaturverzeichnis ... 299

Teil I: Geschichte der Entscheidungstabellentechnik

Die Unzulänglichkeit der seinerzeit angewendeten Hilfsmittel zur Analyse und Beschreibung von Datenverarbeitungsprozessen veranlaßte eine Projektgruppe der General Electric Company, ab 1957 eine neue Beschreibungstechnik zu entwickeln, die "decision structure table" genannt wurde. [1)]

Die Projektgruppe, die im Bereich "Production Control Service" angesiedelt wurde, einem Teil des Hauptbereichs "Manufacturing Services", hatte die nicht unbedeutende Aufgabe, ein integriertes System zu entwickeln, in dem Materialfluß und Informationsfluß weitgehend automatisiert sind.[2)] Schnell stieß die Gruppe auf ihr Hauptproblem: die enorme Anzahl *operativer* Entscheidungen, die täglich getroffen wurden und den Hauptbestandteil des Informationsflusses ausmachten.[3)]

"Operative" Entscheidungen werden, im Gegensatz zu Entscheidungen in der Planungsphase, nicht unter Unsicherheit getroffen. Unter den gleichen Voraussetzungen, d.h. dem gleichen Zustand der Entscheidungsvariablen, führt eine operative Entscheidung auch bei mehrfacher Ausführung immer zum gleichen Resultat.[4)] Zur Beschreibung solcher Entscheidungslogik waren Tabellen und symbolische Ablaufpläne im Gebrauch.

Fertigungsingenieure waren seit längerem an den Gebrauch von Tabellen verschiedenster Art gewöhnt, beispielsweise Tabellen, aus denen sich die für die Fertigung eines Produktes notwendigen Teile ablesen ließen. Solche Tabellen dokumentierten jedoch das Ergebnis der getroffenen Entscheidungen, nicht jedoch die Logik des Entscheidungsprozesses.[5)]

Symbolische Ablaufpläne dagegen waren zwar ein adäquates Hilfsmittel zur Beschreibung sequentieller Prozesse, bei der Beschreibung von komplexen Entscheidungsprozessen mit vielen Variablen und einer großen Anzahl von Handlungsalternativen erfüllten sie jedoch nicht die Anforderungen.[6)]

---

1) Vgl. Kavanagh (TABSOL), S. 117 und S. 120
2) Vgl. Kavanagh (TABSOL), S. 117
3) Vgl. Kavanagh (TABSOL), S. 118
4) Vgl. Kavanagh (TABSOL), S. 118
5) Vgl. Kavanagh (TABSOL), S. 119
6) Vgl. Kavanagh (TABSOL), S. 119 f.

Auf der Basis dieser bereits bekannten Techniken und sicherlich auch unter Nutzung einiger Entwicklungen der mathematischen Logik entwickelte die Projektgruppe mit Unterstützung zahlreicher weiterer Mitarbeiter von General Electric die Entscheidungstabelle. Sie wurde in der Folgezeit mit viel Erfolg vorwiegend bei technischen Problemstellungen, insbesondere bei Konstruktions- und Arbeitsplanungsproblemen, eingesetzt.[1)]

Unbeeinflußt und unabhängig von General Electric wurde die Entscheidungstabelle gleichfalls von Mitarbeitern der Sutherland Company, eines Beratungsunternehmens, entwickelt und seit 1958 eingesetzt.[2)] Aus der zunächst gewählten Bezeichnung "Management rules" wird deutlich, wo der Schwerpunkt ihrer Anwendbarkeit gesehen wurde: in der Dokumentation antizipierter Management-Entscheidungen.[3)] Ihre Bewährungsprobe bestand die zunächst in noch recht rudimentärer Form entwickelte Entscheidungstabellentechnik bei der Systemanalyse für ein komplexes Dateien-Wartungsproblem, das bei Anwendung konventioneller Methoden mit einem Aufwand von 8 bis 10 Mannmonaten nicht gelöst werden konnte. Unter Einsatz von Entscheidungstabellen konnte das Projekt schließlich in zwölf Mannwochen abgeschlossen werden.[4)]

Im Jahr 1959 begann auch Hunt Foods and Industries mit dem Einsatz von Entscheidungstabellen. Und zwar wurden Probleme der kommerziellen Datenverarbeitung mit Hilfe von Entscheidungstabellen analysiert und dokumentiert.[5)]

Ein komplexes Problem der Dateienwartung brachte auch die Insurance Company of North America auf die Spur von Entscheidungstabellen. Ende 1959/Anfang 1960 wurde mit dem Projekt begonnen und nach einer Möglichkeit gesucht, Bedingungen und Aktionen getrennt voneinander zu analysieren und zu beschreiben.[6)] Dabei wurde die Form der Entscheidungsregel als "WENN - DANN" -Verknüpfung von Bedingungen und Aktionen gefunden und wenig später auch die Möglichkeit ihrer tabellarischen Anordnung.[7)]

---

1) Vgl. Kavanagh (Manufacturing Applications), S. 91 ff;
Nickerson (Engineering Application), S. 516 ff.
2) Vgl. Naramore (Application), S. 63
3) Vgl. Naramore (Application}, S. 66
4) Vgl. Naramore (Application), S. 68
5) Vgl. Evans (Advance analysis method) zitiert nach Grad (Tabular form), S. 24
6) Vgl. Brown (Decision table experience), S. 75 f.
7) Vgl. Brown (Decision table experience), S. 77 f.

Von Anfang an verfolgten die meisten der Entwickler und ersten Anwender der Entscheidungstabellentechnik das Ziel, eine formale *Sprache* zu entwickeln, deren wesentliches Element die Entscheidungstabelle war und die sich deshalb besonders gut zur Analyse und Programmierung komplexer Entscheidungen eignen sollte. Aus den Entwicklungsarbeiten von General Electric entstand TABSOL, eine Entscheidungstabellensprache, die bereits 1961 als Bestandteil des umfassenden Kompiliersystems GECOM allen Anwendern der GE225 zur Verfügung gestellt wurde.[1)] Bedingungen und Aktionen innerhalb der Entscheidungstabelle konnten in COBOL formuliert werden, jedoch waren auch einige ALGOL -Elemente zulässig.[2)]
Ebenfalls bei General Electric wurde die auf FORTRAN basierende Entscheidungstabellensprache LOGTAB entwickelt und implementiert,[3)] die jedoch keine größere Bedeutung erlangen konnte.

Mit Unterstützung der IBM entwickelte auch die RAND Corporation Anfang der sechziger Jahre eine auf FORTRAN basierende Entscheidungstabellensprache und erstellte einen Vorübersetzer namens FORTAB. Dieser war als Ergänzung zu FORTRAN gedacht und enthielt nur Sprachelemente, die unbedingt erforderlich waren, um Entscheidungstabellen innerhalb eines FORTRAN-Programms anwenden zu können.[4)]

Die erste Assembler-orientierte Entscheidungstabellensprache entstand bei der Insurance Company of North America. Im Rahmen des bereits erwähnten Projektes wurde LOBOC[5)] entwickelt, eine auf Autocoder basierende Entscheidungstabellensprache, die auch zuerst für die IBM 705 implementiert wurde. [6)]

Ab Anfang 1960 befaßte sich auch die CODASYL (Conference On DAta SYstems Languages) mit der Entwicklung einer Entscheidungstabellensprache. Diese Institution, die 1959 gegründet wurde, um eine problemorientierte, maschinenunabhängige Programmierersprache für kommerzielle Anwendungen zu schaffen, verfolgte das kurzfristige Ziel, eine

---

1) Vgl. Kavanagh (TABSOL), S. 125
2) Vgl. Kavanagh (TABSOL), S. 125 und General Electric (GECOM II), S. 175 ff.
3) Vgl. Cantrell (Applications), S. 56
4) Vgl. Armerding (FORTAB), S. 84 ff.
5) Abkürzung für Logical Business Oriented Coding
6) Vgl. Brown (Decision table experience), S. 78 f. und Devine (Decision tables), S. 461 f.

prozedurorientierte Programmiersprache zu definieren, und das längerfristige Ziel, eine *systemorientierte*[1)] Programmiersprache zu entwickeln, und gründete dazu zwei Komitees.[2)]

Das erste Komitee legte bereits im April 1960 die Erstfassung der COBOL-Sprache "COBOL-60" vor und führte seine Arbeiten zunächst unter der Bezeichnung "COBOL Maintenance Committee" und ab 1964 als "COBOL Committee" weiter.[3)] Das zweite Komitee, das bald die Bezeichnung "Development Committee" erhielt und sich aus zwei Arbeitsgruppen zusammensetzte - "Language Structure Group" und "Systems Group" -, verlagerte innerhalb seiner "Systems Group" die Aktivitäten zur Entwicklung einer systemorientierten Programmiersprache bereits Anfang 1960 auf Entscheidungstabellen als einer möglichen Grundlage für eine solche Sprache. [4)] Da einige der Pioniere der Entscheidungstabellentechnik der "Systems Group" angehörten[5)], gelang es bereits 1962, die vorläufigen Spezifikationen einer Entscheidungstabellensprache für kommerzielle Annwendungen vorzulegen: DETAB-X [6)]. DETAB-X sollte COBOL-61, eine überarbeitete Version von COBOL-60, mit Entscheidungstabellen verbinden. Das Ergebnis war jedoch nicht als Ablösung von COBOL-61 gedacht, sondern als Ergänzung[7)].

---

1) "System" ist in diesem Zusammenhang als "Anwendungssystem" zu interpretieren.
2) Vgl. Calkins (DETAB-X), S. 9
3) Vgl. Standards Committee (Revised X3.23 COBOL), S. XIV-l
4) Vgl. Pollack (CODASYL), S. 60
5) Es waren u.a. vertreten:
   - Burton Grad, ab 1960 Mitarbeiter der IBM, aber bis dahin Leiter der Projektgruppe bei General Electric, die die Entscheidungsstabellentechnik entwickelt hatte. Vgl. Grad (Tabular Form), S. 23 - Orren Y. Evans, der die Entscheidungstabellentechnik bei Hunt Foods and Industries mit entwickelt hatte,
   - Lynn Brown von der Insurance Company of North America, der für die Entwicklung von LOBOC verantwortlich war.
   Chairman der Gruppe war ab 1962 Solomon Pollack von der RAND Corporation, der in den sechziger Jahren etliche methodische Arbeiten über die Entscheidungstabellentechnik vorlegte, die weite Beachtung gefunden haben.
6) Der Zusatz "X" ist die Abkürzung für "Experimental". Vgl. Pollack (DETAB-X), S. 29
7) Vgl. Pollack (DETAB-X), S. 29 und Calkins (DETAB-X), S. 11

In folgenden Punkten wich DETAB-X von COBOL- 61 ab[1)]:

- DETAB-X benutzte ein tabellarisches Format zur Datenbeschreibung innerhalb der DATA DIVISION. Die Datenvereinbarungen wurden an festgelegten Positionen eingetragen. Auch Abkürzungen waren zulässig.

- Bedingungen und Aktionen der in der PROCEDURE DIVISION zugelassenen Entscheidungstabellen wurden in Form von Anweisungen formuliert, die zwar in wesentlichen Bestandteilen mit COBOL-61 übereinstimmten, jedoch sich in vielen Details davon unterschieden. COBOL-Wörter wurden geändert, um sie für eine Verwendung innerhalb der Entscheidungstabelle geeigneter zu machen.

Die ursprüngliche Absicht, DETAB-X nach entsprechender Erprobung in der Praxis in die COBOL-Sprache zu integrieren[2)], konnte nicht verwirklicht werden. 1963 wurde jedoch von der Special Interest Group on Programming Languages (SIGPLAN) des Los Angeles Chapter der Association for Computing Machinery eine Arbeitsgruppe zur Entwicklung eines Vorübersetzers für DETAB-X gegründet. Diese Gruppe revidierte jedoch zunächst die Sprache, um eine maximale Kompatibilität mit COBOL-61 zu erzielen, und entwickelte für die definierte Sprache DETAB/65 auch einen Vorübersetzer gleichen Namens, der kostenlos an interessierte DV-Anwender abgegeben wurde. [3)] Der DETAB/65-Vorübersetzer, der jede eingegebene Entscheidungstabelle, deren Bedingungen und Aktionen in COBOL formuliert waren, in ein COBOL-Kapitel umwandelte, umfasste jedoch nicht den vollen DETAB/65-Sprachumfang[4)] und konnte insbesondere wegen völlig fehlender Optimierungseigenschaften und völligem Verzicht auf Wartung in den Folgejahren keinen Durchbruch einer rechnergestützten Anwendung der Entscheidungstabellentechnik bewirken. [5)]

---

1) Vgl. zum folgenden
Pollack (DETAB-X), S. 31 ff;
Calkins (Question and answer), S. 46 ff.; Pollack, Wright (Data description), S. 2 ff.

2) Vgl. Pollack (CODASYL), S. 61

3) Vgl. SIGPLAN (DETAB/65), S. 1 f.
Zum DETAB/65-Vorübersetzer vgl. auch Callahan, Chapman (DETAB/65 pre-processors), S. 441 ff.

4) Beispielsweise wurden nur begrenzte Entscheidungstabellen verarbeitet.

5) Vgl. Pollack; Hicks; Harrison (Decision tables), S. 5

Möglicherweise hat der DETAB/65-Vorübersetzer jedoch gerade wegen seiner fehlenden Optimierungseigenschaften Softwarehäuser und Datenverarbeitungsanwender in den Folgejahren herausgefordert, bessere Produkte zu schaffen. Seit 1966 entstanden zunächst in den USA, später auch in Europa, Dutzende von Entscheidungstabellenvorübersetzern, von denen ein großer Teil auch gegenwärtig noch im Einsatz ist. [1)]

Wie der ausführliche historische Rückblick gezeigt hat, lässt sich nach der anfänglichen Entwicklung und Definition ihrer wichtigsten Bestandteile Ende der fünfziger, Anfang der sechziger Jahre keinesfalls ein Siegeszug der Entscheidungstabellentechnik konstatieren. Einer der Pioniere in der Entwicklung der Technik, Burton Grad, beschreibt in einem 1971 veröffentlichten Rückblick seine Gefühle und die seiner "Leidensgenossen" wie folgt:

"Throughout 1963 we had the pleasure of creating, defining, and proselytizing. Since 1963 we have had the frustration of finding that business analysts, system engineers, and computer programmers did not immediately accept decision tables as a major technique to assist in program design, implementation, and maintenance. "[2)]

Verschiedene Anzeichen deuten darauf hin, dass das Interesse der Datenverarbeitungspraxis an der Entscheidungstabellentechnik in den letzten Jahren seit 1970 wieder deutlich gewachsen ist. Nachdem zunächst Hardware-Hersteller und Softwarehäuser Erfahrungen mit der Entwicklung komplexer Softwaresysteme für Datenverarbeitungsanlagen der sogenannten dritten Generation sammeln konnten, begannen auch Anwender mit der Entwicklung komplexer Anwendungssysteme, die leicht einen Aufwand von 20 Mannjahren erreichten und auch überstiegen. Die zum Teil bitteren Erfahrungen, die bei der Entwicklung dieser Systeme gemacht wurden, ließen Zweifel an den herkömmlichen Methoden der Systemplanung und -implementierung aufkommen. Die "Reliability" (also die *Zuverlässigkeit*) der Programme geriet unversehens in den Brennpunkt des Interesses und damit alle Methoden, deren Anwendung versprach, die Zuverlässigkeit der Programme zu erhöhen.

---

1) Folgende Quellen geben einen Überblick über die bis etwa 1970 entstandenen Vorübersetzer und vergleichen zum Teil auch die Fähigkeiten der Produkte:
NCC (Decision tables), S. 65 ff;
McDaniel (Software), S. 1 ff;
o. V. (COBOL support packages), S. 7 ff;
Strunz (Voruntersuchungen), S. 32 ff.

2) Pollack; Hicks; Harrison (Decision tables), S. VII (Foreword)

Auch die Wirksamkeit allgemein angewendeter Methoden zur Verifikation von Programmen geriet in Zweifel. Die Problematik wird durch nichts besser beleuchtet als durch einen mittlerweile berühmt gewordenen Ausspruch Dijkstras: "Program testing can be used to show the presence of bugs, but never to show their absence.„ [1)]

Schließlich trat auch die Erkenntnis in das allgemeine Bewusstsein, dass Hardware-Kosten tendenziell sinken, die Kosten der Entwicklung von Anwendungssystemen aufgrund ihres hohen Anteils an Personalkosten jedoch stark steigen. Zusammen mit einem beispiellosen Aufschwung in der Anwendung problemorientierter Programmiersprachen, die eine Erhöhung der Programmiererproduktivität ermöglichen, gewannen nun auch produktivitätsfördernde Programmiermethoden eine wachsende Zahl von Anwendern.

Eine Reihe dieser Methoden befasst sich mit *strukturellen* Eigenschaften der Programme und abstrahiert von ihrem spezifischen Inhalt. So versucht die *Modulare Programmierung*, komplexe Strukturen dadurch zu vermeiden, dass sie große Aufgabenkomplexe in kleinere Einheiten unterteilt, deren untereinander bestehende Beziehungen minimiert werden. [2)] Die *Strukturierte Programmierung* enthält unter anderem eine Vorschrift zur Gestaltung der Programmcodierung. Nur einige wohldefinierte, elementare Ablaufstrukturen sind zugelassen, aus denen jedes Programm aufgebaut werden soll. [3)]

Untersuchungsgegenstand der vorliegenden Arbeit ist die Entscheidungsstabellentechnik in ihrer Eigenschaft als *Strukturs*prache zur Gestaltung von Entscheidungsprozessen innerhalb rechnergestützter Informationssysteme. Die Syntax, Semantik und Pragmatik dieser Strukturs*prache* sollen weitgehend losgelöst von den Eigenschaften anderer in der Systementwicklung in Kombination mit Entscheidungstabellen benutzten natürlichen oder formalen Sprachen behandelt werden, um eine Beurteilung ihrer Leistungsfähigkeit zu ermöglichen.

Bei der Diskussion ihrer pragmatischen Dimension wird die Darstellung der Einsatzmöglichkeiten der Entscheidungstabellentechnik zur Gestaltung rechnergestützter Informationssysteme im Vordergrund stehen. Einen Schwerpunkt soll dabei die Erörterung der Frage bilden, inwieweit

---

1) Dijkstra (Structured programming), S. 85
2) Vgl. dazu Abschnitt 5.1.2.1
3) Vgl. dazu Abschnitt 5.1.2.2.1.1

die Entscheidungstabellentechnik selbst *rechnergestützt* angewendet werden kann.

Teil II: Theoretische Grundlagen der Entscheidungstabellentechnik

1. Theorie der Eintreffer-Entscheidungstabelle

Als "Eintreffer -Entscheidungstabelle" soll die historisch zuerst entwickelte, im Schrifttum fast ausschließlich dargestellte und deshalb auch in der Praxis meist verbreitete Entscheidungstabellenart bezeichnet werden, deren Regeln echte Verarbeitungsalternativen kennzeichnen. Die anderen Erstellungs- und Interpretationsregeln unterliegende "Mehrtreffer -Entscheidungstabelle" wird an anderer Stelle ausführlich diskutiert.[1)]

Dem Untersuchungsziel entsprechend soll die Eintreffertabelle in ihrer Funktion als *Sprache* zur Beschreibung struktureller Aspekte von Entscheidungen untersucht werden.

Nach einem der Begründer der Semiotik, Charles Morris[2)], kann jede Sprache als System von Zeichen verstanden werden, die untereinander in Beziehung stehen. [3)] Jedes Zeichen kann in den drei Dimensionen Syntax, Semantik und Pragmatik untersucht werden. [4)]

Die *Pragmatik* untersucht die Beziehung eines Zeichens zu demjenigen, der es gebraucht;
die *Semantik* untersucht die Beziehung eines Zeichens zu dem Objekt, das es bezeichnet;
die *Syntax* untersucht die formale Beziehung des Zeichens zu anderen Zeichen.

Diese drei Dimensionen stehen untereinander in einem bestimmten Verhältnis. Die pragmatische Beziehung setzt die semantische und syntaktische Beziehung voraus, die semantische Beziehung die syntaktische. Umgekehrt kann die Syntax eines Zeichens betrachtet werden, ohne dass Semantik und Pragmatik berücksichtigt werden. Semantische Untersuchungen sind wiederum ohne Betrachtung der Pragmatik möglich.

Sprachen sind besondere Systeme von Zeichen, die eine syntaktische Struktur derart haben, dass unter den erlaubten Zeichenkombinationen einige als Ausdrücke fungieren können und nur Zeichen benutzt werden, deren Bedeutung sich mehreren Sprachbenutzern im gleichen Sinn erschließt.

---

1) Vgl. dazu Abschnitt 2
2) Als grundlegendes Werk der Semiotik gilt Morris (Theory of signs)
3) Vgl. Morris (Theory of signs), S. 11
4) Vgl. Morris (Theory of signs), S. 6 f.

Die Anwendung einer Sprache erfordert die Beherrschung der syntaktischen, semantischen und pragmatischen Regeln dieser Sprache.[1)]

Es lassen sich universelle und spezielle Sprachen unterscheiden.[2)]

Mit *universellen* Sprachen lassen sich alle Sachverhalte ausdrücken. Die natürlichen Sprachen wie Deutsch, Englisch und Französisch usw. gelten als universelle Sprachen. Solche Sprachen haben die Eigenschaft einer ausgeprägten Redundanz, die einerseits eine flexible Anwendung der Sprache ermöglicht, andererseits jedoch auch die Ursache für Missverständnisse in der pragmatischen Ebene der Sprachbenutzung sein kann.

Spezielle Sprachen sind eingeschränkte Sprachen, die nur für bestimmte Zwecke benutzt werden. Wenn sie auf der syntaktischen Ebene so eindeutig definiert sind, dass ein Operieren mit der Sprache möglich ist, ohne dass die "Bedeutung" der Zeichen, also die semantische Ebene, berührt ist, kennzeichnet man diese Sprache als "formalisiert"[3)] oder "formal". Syntaxregeln einer formalen Sprache umfassen Regeln, die bestimmen, welche Zeichen und Zeichenkombinationen zulässig sind, sowie Regeln, die bestimmen, wie Ausdrücke gebildet werden können.[4)] Operationen mit Elementen einer Sprache auf syntaktischer Ebene sind jedoch nur dann sinnvoll, wenn die Ergebnisse semantisch interpretierbar sind.

1.1 Syntax der Entscheidungstabelle

Die *Syntax* der Entscheidungstabelle umfasst die Definition ihrer elementaren syntaktischen Einheiten, die Regeln zur Bildung höherer syntaktischer Einheiten und die Transformationsregeln, nach denen syntaktische Einheiten aus anderen syntaktischen Einheiten abgeleitet werden. Ergänzend sollen Verfahren erläutert werden, die die korrekte Anwendung von Bildungs- und Transformationsregeln überprüfen.

1.1.1 Bildungsregeln

Die Bildungsregeln einer Sprache umfassen jenen Teil der syntaktischen Regeln, der sich mit dem Aufbau höherer syntaktischer Einheiten aus elementaren syntaktischen Einheiten befasst. Auf der Basis der Definition der jeweiligen elementaren syntaktischen Einheiten werden alle Regeln formuliert, die zu den erlaubten höheren syntaktischen Einheiten führen.

---

1) Vgl. Morris (Theory of signs), S. 11
2) Vgl. Morris (Theory of signs), S. 11 f.
3) Vgl. Bochenski (Denkmethoden), S. 42
4) Vgl. S. Bochenski (Denkmethoden), S. 42 f.

1.1.1.1 Elementare syntaktische Einheiten

Elementare syntaktische Einheiten der Entscheidungstabelle sind Bedingungen mit Bedingungsanzeigern und Aktionen mit Aktionsanzeigern.

1.1.1.1.1 Bedingungen und Bedingungsanzeiger

Zunächst sollen Bedingungen und Bedingungsanzeiger definiert werden.

Def.: Die *Bedingungen B* als elementare syntaktische Einheiten einer Entscheidungstabelle sind Variablen mit einer endlichen Anzahl möglicher Ausprägungen; die Bedingungen werden mit den Symbolen a, b, c, ... usw. bezeichnet.

Die zu den Bedingungen gehörigen Mengen der Ausprägungen werden mit [a], [b], [c], usw. bezeichnet. Die Ausprägungen werden *Bedingungsanzeiger*[1)] genannt. Für sie werden die Symbole a1, a2, ... , am; b1, b2, ... , bn; c 1, c2, c3 ...,
ck; usw. gewählt.

Für den Fall, dass eine Bedingung nur zwei Ausprägungen hat, kann die Form

Bedingung a mit [a] = {J, N} benutzt werden. In diesem Fall wird bereits durch den Ausdruck "Bedingung a" eine der beiden Ausprägungen explizit beschrieben und durch die Anzeiger bejaht oder verneint.

Eine Bedingung in der zuletzt angegebenen Schreibweise wird "einfache Bedingung" genannt, in der zuerst angegebenen Schreibweise wird sie als "komplexe Bedingung" bezeichnet.

Für die Betrachtung der Entscheidungstabelle in ihrer syntaktischen Dimension reicht diese Definition der Bedingung als elementare syntaktische Einheit völlig aus. In dieser Form übernimmt sie Platzhalterfunktion für die Beschreibung problembezogener Bedingungen.

---

1) Diese sinngemäße Übersetzung für das englische "condition entries" wurde vom Autor 1970 vorgeschlagen und im seither erschienenen deutschsprachigen Schrifttum weitgehend akzeptiert. Vgl. Strunz (Entscheidungstabellen), S. 58

Hat eine Bedingung eine große Anzahl verschiedener Ausprägungen, die jedoch nicht alle explizit dargestellt zu werden brauchen, dann kann ein ELSE -Bedingungsanzeiger zur Repräsentation einer beliebig großen Restgruppe von Bedingungsausprägungen benutzt werden.[1)]

Def.: Werden nicht alle Ausprägungen einer Bedingung x explizit angegeben, dann kann der *ELSE -Bedingungsanzeiger* ε (x) benutzt werden. Er bezeichnet die Ausprägungen, die nicht durch andere Bedingungsanzeiger explizit beschrieben sind.

Beispiel;

Bedingung x mit [x] = {xl, x2, ... , xn}

soll insgesamt 29 Ausprägungen aufweisen, von denen jedoch nur 5 gesondert behandelt werden müssen und die restlichen 24 Ausprägungen zu einem Ausdruck zusammengefasst werden können. Dann ist folgende Formulierung zulässig:

Bedingung x mit [x] = {xl, x2, x3,x4,x5, ε}

1.1.1.1.2 Aktionen und Aktionsanzeiger

Für Aktionen und Aktionsanzeiger gilt die folgende Definition.

Def.: Die Aktionen A als elementare syntaktische Einheiten einer Entscheidungstabelle sind Variablen mit einer endlichen Anzahl möglicher Ausprägungen; die Aktionen werden mit den Symbolen α, β, γ, ... usw. bezeichnet.

Die zu den Aktionen gehörigen Mengen der Ausprägungen werden mit **[α]**, **[β]**, **[γ]**,... usw. bezeichnet. Die Ausprägungen werden Aktionsanzeiger genannt. Für sie werden die Symbole α1, α2,... , αm; β1, β2, ... , βn; γl, γ2, γk; ... usw. gewählt.
Für den Fall, dass eine Aktion nur eine Ausprägung hat, kann die Form

Aktion a mit [α] = {x}

benutzt werden. In dieser Schreibweise wird eine Aktion als "einfache Aktion" bezeichnet, in der zuerst angegebenen Schreibweise als "komplexe Aktion".

In der angegebenen Schreibweise übernehmen die benutzten syntaktischen Formen Platzhalterfunktionen für die Beschreibung problembezogener Aktionen.

---

1) Ein ELSE -Bedingungsanzeiger in Form des Sprachelements "NOT" war als Bestandteil der DETAB/65-Sprache definiert worden. Vgl. SIGPLAN (DETAB/65), S. 19 f .. Siehe auch Pollack (Decision tables), S. 64. Woods und Hawes schlagen "NNN" als ELSE -Bedingungsanzeiger vor. Vgl. Woods; Hawes (Code generation), S. 75 f.

1.1.1.2 Höhere syntaktische Einheiten

Aus den elementaren syntaktischen Einheiten lassen sich die Basisfolge von Bedingungen und Aktionen sowie Bedingungs- und Aktionsanzeigerkombinationen bilden. Durch Zuordnung einer Aktionsanzeigerkombination zu einer Bedingungsanzeigerkombination entsteht die Entscheidungsregel.

Als Zusammenfassung je einer Basisfolge von Bedingungen und Aktionen sowie von Entscheidungsregeln entsteht die Entscheidungstabelle. Mehrere Entscheidungstabellen schließlich können zur höchsten syntaktischen Einheit, dem Entscheidungstabellenverbund, zusammengefasst werden.

1.1.1.2.1 Basisfolge der Bedingungen

Alle Bedingungen einer Entscheidungstabelle können als höhere syntaktische Einheit betrachtet werden.

Def.: Die in einer Entscheidungstabelle aufgeführten Bedingungen bilden die *Basisfolge der Bedingungen* dieser Entscheidungsstabelle.

Die Bedingungen in einer Basisfolge sind geordnet, so dass die Reihenfolge der Bedingungsprüfungen jeder aus der Basisfolge abgeleiteten Bedingungsfolge klar definiert ist.

1.1.1.2.2 Basisfolge der Aktionen

Auch die Gesamtheit der Aktionen einer Entscheidungstabelle bildet eine höhere syntaktische Einheit.

Def.: Die in einer Entscheidungstabelle aufgeführten Aktionen bilden die *Basisfolge der Aktionen* dieser Entscheidungstabelle.

Die Aktionen in einer Basisfolge sind geordnet, so dass die Reihenfolge der Aktionen einer aus der Basisfolge abgeleiteten Aktionsfolge klar definiert ist.

1.1.1.2.3 Bedingungsanzeigerkombinationen

Durch Kombination von Bedingungsanzeigern lässt sich eine weitere höhere syntaktische Einheit bilden.

Def.: Werden je ein Bedingungsanzeiger verschiedener Bedingungen einer Basisfolge miteinander kombiniert, dann entsteht eine *Bedingungsanzeigerkombination*.

Für die drei Bedingungen a, b und c in der Basisfolge

Bedingung a mit [a] = {a1,a2,a3}
Bedingung b mit [b] = {b1,b2}
Bedingung c mit [c] = {J,N}

lassen sich beispielsweise folgende Bedingungsanzeigerkombinationen bilden: [1)]

1	2	3	4	5	6	7	8
a1	a1	a1	a2	a2	a3	a3	a3
b1	b1	b2	b1	b2	b1	b1	b2
J	N	N	J	J	J	N	J

1.1.1.2.4 Aktionsanzeigerkombinationen

Auch aus Aktionsanzeigern lässt sich eine höhere syntaktische Einheit bilden.

Def.: Eine *Aktionsanzeigerkombination* entsteht aus der Kombination je eines Anzeigers verschiedener Aktionen einer Basisfolge.

Für die drei Aktionen α, β und γ in der angegebenen Basisfolge

Aktion α mit $[\alpha] = \{\alpha 1,\alpha 2,\alpha 3,\alpha 4\}$
Aktion β mit $[\beta] = \{X\}$
Aktion γ mit $[\gamma] = \{\gamma 1,\gamma 2\}$

lassen sich beispielsweise folgende Aktionsanzeigerkombinationen bilden:

1	2	3	4	5
α1	α1	α2	α3	α4
X	X	X	X	X
γ1	γ2	γ2	γ2	γ2

1) Nur ein Teil aller formal möglichen Kombinationen ist beispielhaft wiedergegeben.

1.1.1.2.5 Entscheidungsregeln

Aus bereits definierten höheren syntaktischen Einheiten lässt sich eine weitere syntaktische Einheit bilden.

Def.: Die *Entscheidungsregel* entsteht aus der Zuordnung einer Aktionsanzeigerkombination zu einer Bedingungsanzeigerkombination.

Beispielsweise lassen sich aus einigen der in den vorigen Abschnitten gebildeten Bedingungs- und Aktionsanzeigerkombinationen folgende Entscheidungsregeln bilden:

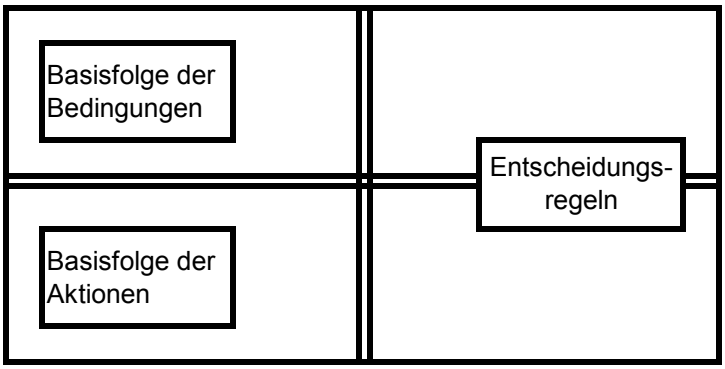
1	2	3	4	5	6
a1	a1	a1	a2	a2	a3
b1	b1	b1	b1	b1	b2
J	J	N	J	J	J
$\alpha 1$	$\alpha 1$	$\alpha 2$	$\alpha 3$	$\alpha 4$	$\alpha 5$
X	X	X	X	X	X
$\gamma 1$	$\gamma 2$	$\gamma 1$	$\gamma 2$	$\gamma 2$	$\gamma 2$

1.1.1.2.6 Entscheidungstabellen

Die bisher definierten syntaktischen Einheiten erlauben nunmehr auch eine Definition der Entscheidungstabelle.

Def.: Die *Entscheidungstabelle* besteht aus je einer Basisfolge von Bedingungen und Aktionen sowie aus Entscheidungsregeln, deren Bedingungs- und Aktionsanzeigerkombinationen aus Anzeigern der in den Basisfolgen enthaltenen Bedingungen und Aktionen bestehen.

Nur mit Hilfe einer zweidimensionalen Notation lässt sich diese syntaktische Einheit darstellen. Bei vertikaler Anordnung der Entscheidungsregeln ergibt sich folgendes Darstellungsformat:



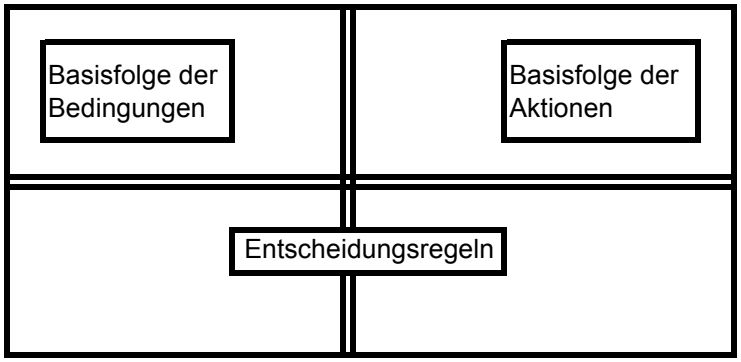
Um eine übersichtliche Darstellung zu erhalten, sind Bedingungen, Aktionen und die Anzeigerkombinationen durch Doppelstriche voneinander getrennt. Zu den bisher als Beispiel verwendeten elementaren und höheren syntaktischen Einheiten kann die folgende Entscheidungstabelle im vertikalen Format gebildet werden:

Bedingung a mit [a] = {a1,a2,a3}	a1	a2	a2	a3	a3	a3
Bedingung b mit [b] = {b1,b2}	b1	b2	b2	b1	b2	b2
Bedingung c mit [c] = {J,N}	J	J	N	N	J	N
Aktion α mit [α] = {α1,α2,α3,α4}	α4	α2	α1	α1	α4	α3
Aktion β mit [β] = {X}	X	X	X	X	X	X
Aktion γ mit [γ] = {γ1,γ2}	γ1	γ2	γ1	γ2	γ1	γ1

Bei horizontaler Anordnung der Entscheidungsregeln ergibt sich folgendes Format:[1)]

---

1) Das horizontale Format ist das historisch zuerst entwickelte Format.
Vgl. Kavanagh (TABSOL), S. 120 f.



Die Beispieltabelle ergibt sich in diesem Format wie folgt:

Bedingung a mit [a] = {a1,a2,a3}	Bedingung b mit [b] = {b1,b2}	Bedingung c mit [c] = {J,N}	Aktion α mit [α] = {α1,α2,α3,α4}	Aktion β mit [β] = {X}	Aktion γ mit [γ] = {γ1,γ2}
a1	b1	J	α4	X	γ1
a2	b2	J	α2	X	γ2
a2	b2	N	α1	X	γ1
a3	b1	N	α1	X	γ2
a3	b2	J	α4	X	γ1
a3	b2	N	α3	X	γ1

Da der Platzbedarf für die Basisfolgen der Bedingungen und Aktionen im allgemeinen erheblich größer ist als für die Anzeiger, eignet sich das vertikale Format besser zur Darstellung der Entscheidungstabelle. Es hat sich international durchgesetzt und soll deshalb in Übereinstimmung mit Pollack, Hicks und Harrison als "Standardformat"[1)] bezeichnet werden. Unter den derzeit bekannten, bedeutenden Anwendern der Entscheidungs-

1) Pollack, Hicks und Harrison führen die Bevorzugung des vertikalen Formats auf die Tätigkeit der CODASYL Systems Group und auf die Ergebnisse des von ihr 1962 in New York ausgerichteten Symposiums zurück. Vgl. Pollack, Hicks, Harrison (Decision tables), S. 5

tabellentechnik verwenden nur noch wenige das horizontale Format. [1)]

In der Praxis der Anwendung der Entscheidungstabellentechnik haben sich folgende Änderungen und Ergänzungen in der Notation durchgesetzt:

Es ist nicht üblich, im linken oberen und linken unteren Quadranten der Tabelle die Bedingungs- und Aktionsanzeiger explizit aufzuführen. Die Praxis geht offensichtlich davon aus, dass die Bedingungsanzeiger und Aktionsanzeiger innerhalb der Entscheidungsregeln ausgewiesen werden, so dass die Bedingungen und Aktionen als elementare syntaktische Einheiten der Entscheidungstabelle durch Bedingungs-/Aktions- und Regelteil der Tabelle definiert sind.

Zur besseren Lesbarkeit der Entscheidungstabelle können den vier Quadranten Identifikationsteil, Regelbezeichner 1,2,3, ... ,k
Bedingungsbezeichner B1, B2, B3,……..,Bm, Aktionsbezeichner
AI, A2, A3,……..An, und ein Bemerkungsteil hinzugefügt werden.[2)]
Die folgende Abbildung illustriert die Ergänzungselemente am Beispiel der bisher benutzten Entscheidungstabelle.

Beispieltabelle		1	2	3	4	5	6
B1	Bedingung a	a1	a2	a2	a3	a3	a3
B2	Bedingung b	b1	b2	b2	b1	b2	b2
B3	Bedingung c	J	J	N	N	J	N
A1	Aktion α	α4	α2	α1	α1	α4	α3
A2	Aktion β	X	X	X	X	X	X
A3	Aktion γ	γ1	γ2	γ1	γ2	γ1	γ1
Bemerkungen:							

1) Beispielsweise verwendet die US Air Force das horizontale Format wahlweise zum vertikalen Format. Vgl. Air Force (Pamphlet 5-5), S. 7
Die B. F. Goodrich Chemical Company, Cleveland, Ohio, ist ein weiteres Beispiel. Seit 1962, als dort die Entscheidungstabellentechnik unter Einsatz des Entscheidungstabellenvorübersetzers TABSOL eingeführt wurde, wird mit dem horizontalen Format gearbeitet. Vgl. Sterbenz (TABSOL), S. 33

2) Zu den Ergänzungselementen einer Entscheidungstabelle vgl. IBM (Decision tables), S. 19

Nicht immer beeinflussen alle Bedingungen einer Basisfolge gemeinsam die Auswahl der Aktionsfolgen. Ebenso bestehen nicht alle auszuführenden Aktionsfolgen immer aus allen Aktionen der Basisfolge. Die bisher definierten syntaktischen Möglichkeiten können diesen Anforderungen noch nicht gerecht werden, da das zweidimensionale Format der Entscheidungstabelle innerhalb der Regeln Kombinationen aus allen Anzeigern der enthaltenen Bedingungen und Aktionen erfordert. Durch die Definition zusätzlicher Anzeigerformen lässt sich dieses Problem jedoch syntaktisch lösen, ohne das Entscheidungstabellenformat ändern zu müssen.

Def. : Der *Ausschlußanzeiger* "#" [1] innerhalb einer Bedingungsanzeigerkombination und "~" (Leerstelle) innerhalb einer Aktionsanzeigerkombination einer Entscheidungsregel schließt die tabellarisch zugeordnete Bedingung oder Aktion der Basisfolge aus der angezeigten Folge von Bedingungen oder Aktionen aus.

Zusätzlich ist es semantisch sinnvoll, zwischen der Tatsache, dass eine Bedingung nicht Bestandteil einer Bedingungsfolge ist, und der Tatsache, dass sie zwar Bestandteil ist, aber die innerhalb einer Entscheidungsregel zugeordnete Aktionsfolge nicht beeinflusst, zu differenzieren. Die Syntax der Entscheidungstabelle wird deshalb um folgende Definition erweitert:

Def.: Der "*Indifferenzanzeiger* "–" [2] innerhalb der Bedingungsanzeigerkombination einer Entscheidungsregel steht als Symbol für die Menge aller expliziten Bedingungsanzeiger der tabellarisch zugeordneten Bedingung.

---

1) Der Ausschlußanzeiger innerhalb einer Bedingungsanzeigerkombination ist in der bisherigen Entscheidungstabellenliteratur nicht bekannt. Erfordernisse der Praxis führten in der Decision Tables Task Group der CODASYL dazu, den Ausschlußanzeiger auch innerhalb einer Bedingungsanzeigerkombination zuzulassen.
Vgl. Theory subgroup, (DT glossary), S. 8
Der Ausschlußanzeiger in einer Bedingungsanzeigerkombination wurde erstmalig bei dem ab 1973 in der Praxis eingesetzten Entscheidungstabellen -Vorübersetzer VORELLE zugelassen.
Vgl. mbp (VORELLE-Handbuch), S. 2 - 11

2) In der Entscheidungstabellenliteratur hat der Indifferenzanzeiger eine Fülle von Bezeichnungen, z.B. "don't care", "dash", "Nichtrelevant-Anzeiger", "Irrelevanzanzeiger". Gildersleeve nannte diesen Anzeiger sehr zutreffend "indifference entry". Dieser Bezeichnung soll hier gefolgt werden.
Vgl. Gildersleeve (Decision tables), S. 59

Ausschlußanzeiger und Indifferenzanzeiger können auch unter dem gemeinsamen Begriff Irrevelanzanzeiger zusammengefasst werden.[1]

1.1.1.2.7 Entscheidungstabellenverbund

Aus den höheren syntaktischen Einheiten "Entscheidungstabellen" lässt sich eine weitere syntaktische Einheit bilden.

Def.: Mehrere untereinander durch Verknüpfungsanweisungen logisch verbundene Entscheidungstabellen bilden einen *Entscheidungstabellenverbund*.[2)]

Die Notwendigkeit, den Entscheidungstabellenverbund als weitere syntaktische Einheit einzuführen, resultiert daraus, dass die Anzahl möglicher Bedingungsanzeigerkombinationen mit zunehmender Anzahl relevanter Bedingungen stark überproportional anwächst. Bei 5 Bedingungen mit je 2 Bedingungsanzeigern können bereits $2^5 = 32$ unterschiedliche Bedingungsanzeigerkombinationen gebildet werden. Bei 6 Bedingungen wären $2^6 = 64$, bei 7 Bedingungen $2^7 = 128$ usw. Bedingungsanzeigerkombinationen möglich. Auch wenn in vielen Fällen nicht alle Bedingungsanzeigerkombinationen innerhalb der Entscheidungsregeln explizit dargestellt werden, sondern durch Benutzung von Ausschluss- und Indifferenzanzeigern die Anzahl der Regeln erheblich unter der Anzahl der maximal möglichen Bedingungsanzeigerkombinationen liegen kann, werden die Grenzen einer übersichtlichen tabellarischen Darstellbarkeit mit zunehmender Anzahl von Bedingungen rasch erreicht.

Wenn sich innerhalb eines komplexen Entscheidungsproblems Gruppen von Bedingungen und Aktionen ausgliedern lassen, können mehrere kleinere Entscheidungstabellen erstellt werden. Syntaktisch ergibt sich dann die Notwendigkeit, mehrere Entscheidungstabellen zu einem Entscheidungstabellenverbund zusammenzufassen. Die Verbindungen zwischen den Tabellen werden durch besondere Aktionen geschaffen, die einen sequentiellen Übergang zu einer anderen Entscheidungstabelle ohne Rücksprung oder einen vorübergehenden Übergang mit Rücksprung definieren.

---

1) Der DIN-Normentwurf 66241 kennt nur Irrelevanzanzeiger. Er verzichtet auf die genaueren Bezeichnungen Ausschlußanzeiger und Indifferenzanzeiger.
Vgl. FNI (Entscheidungstabelle), 5.4

2) In der englischsprachigen Entscheidungstabellen-Literatur wird diese syntaktische Einheit vielfach "decision tables network" genannt. Neben "Entscheidungstabellenverbund" ließe sich zu ihrer Bezeichnung demnach auch der Begriff " Entscheidungstabellennetzwerk" verwenden.

Beispiel:

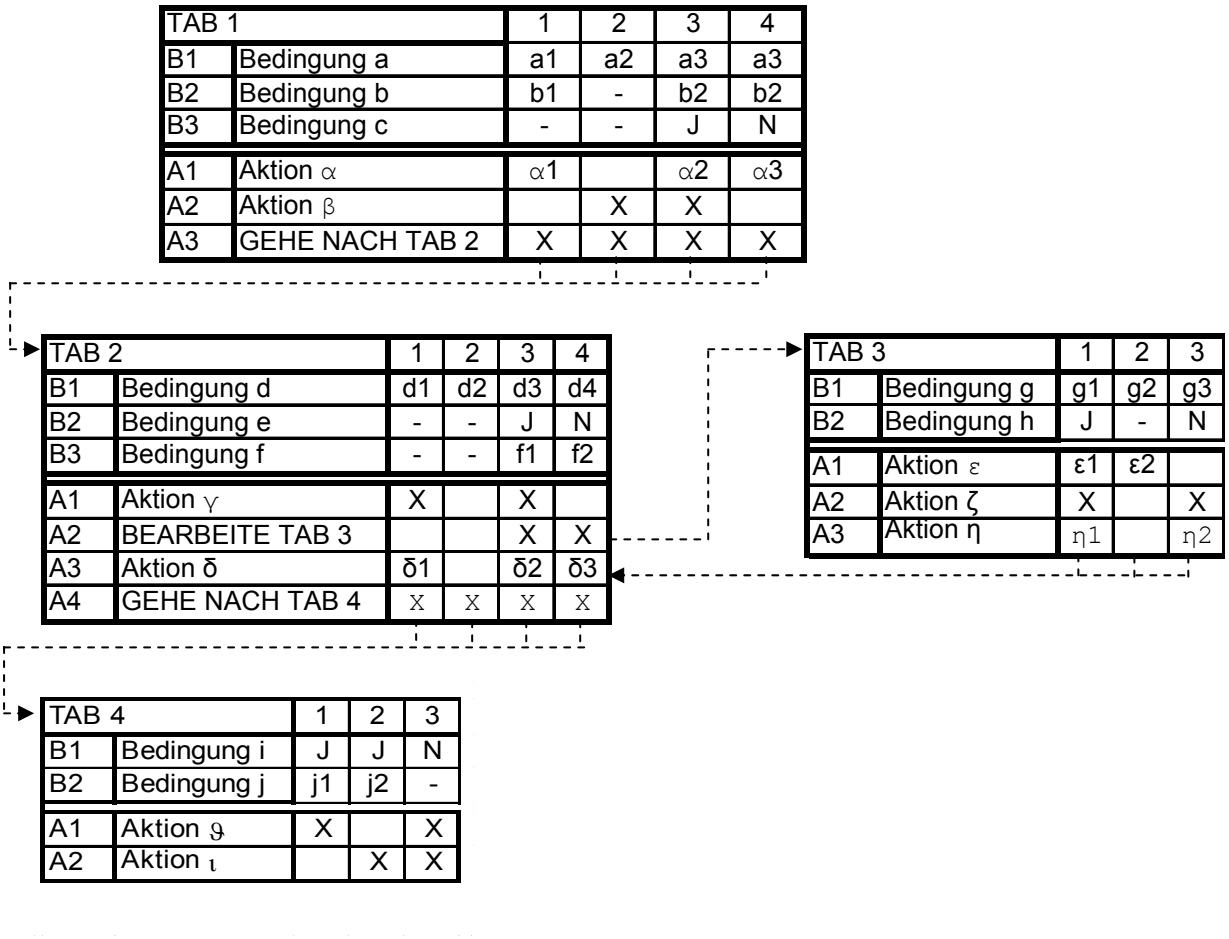

TAB 1 | 1 | 2 | 3 | 4
B1 | Bedingung a | a1 | a2 | a3 | a3
B2 | Bedingung b | b1 | - | b2 | b2
B3 | Bedingung c | - | - | J | N
A1 | Aktion α | α1 | | α2 | α3
A2 | Aktion β | | X | X |
A3 | GEHE NACH TAB 2 | X | X | X | X
TAB 2 | 1 | 2 | 3 | 4
B1 | Bedingung d | d1 | d2 | d3 | d4
B2 | Bedingung e | - | - | J | N
B3 | Bedingung f | - | - | f1 | f2
A1 | Aktion γ | X | | X |
A2 | BEARBEITE TAB 3 | | | X | X
A3 | Aktion δ | δ1 | | δ2 | δ3
A4 | GEHE NACH TAB 4 | X | X | X | X
TAB 3 | 1 | 2 | 3
B1 | Bedingung g | g1 | g2 | g3
B2 | Bedingung h | J | - | N
A1 | Aktion ε | ε1 | ε2 |
A2 | Aktion ζ | X | | X
A3 | Aktion η | η1 | | η2
TAB 4 | 1 | 2 | 3
B1 | Bedingung i | J | J | N
B2 | Bedingung j | j1 | j2 | -
A1 | Aktion ϑ | X | | X
A2 | Aktion ι | | X | X

Abb. 1: Entscheidungstabellenverbund

Zur Veranschaulichung der prozeduralen Beziehungen zwischen den Entscheidungstabellen eines Entscheidungstabellenverbunds können alle Darstellungsformen eines gerichteten Graphen benutzt werden.[1)]

In der Praxis weit verbreitet sind Entscheidungstabellen-Diagramme[2)], in denen Rechtecksymbole als Knoten die einzelnen Entscheidungstabellen repräsentieren und gerichtete Kanten die prozeduralen Verknüpfungen.

In der Umgebung einer interaktiven rechnergestützten Erstellung von Entscheidungstabellen [3)] könnte eine Beschreibung über Formeln oder Matrizen sinnvoller sein.

Ein Entscheidungstabellenverbund V lässt sich darstellen als V = {T, P}, wobei T die Menge aller enthaltenen Entscheidungstabellen t_i (i = 1, ... ,n) ist und P die Menge aller prozeduralen Verknüpfungen P_j (j = I,….m) zwischen jeweils zwei Tabellen des Verbunds. Dabei können diese Beziehungen in der Form

$$P_j = <t_k; t_l>$$

dargestellt werden.

Ein Entscheidungstabellenverbund lässt sich in Form einer quadratischen Matrix mit den einzelnen Entscheidungstabellenindizes als Zeilen- und Spaltenbezeichnung beschreiben. Die prozeduralen Verknüpfungen innerhalb der Matrix werden durch "1" angezeigt.

von \ nach	t1	t2	t3	t4	t5	.....
t1		1		1		
t2			1			
t3	1			1	1	
t4		1				
t5	1			1		
:						

1) Vgl. Strunz (Decision table networks), S. 6 ff.
2) Vgl. dazu beispielsweise London (Decision tables), S. 87 f.
3) Vgl. dazu Abschnitt 4.3 dieser Arbeit

1.1.1.3 Bildung höherer syntaktischer Einheiten

Der Prozess der Entscheidungstabellenerstellung kann auf Syntax-Ebene als Bildung höherer syntaktischer Einheiten aus elementaren syntaktischen Einheiten verstanden werden. Die Bildung dieser höheren syntaktischen Einheiten unterliegt den im Folgenden zu erläuternden Regeln.

1.1.1.3.1 Bildung einer Basisfolge von Bedingungen

Zunächst soll die Bildung einer Basisfolge von Bedingungen erläutert werden.

Def.: Eine Basisfolge aus n Bedingungen entsteht durch Anordnung dieser n Bedingungen in einer Reihenfolge.

Die Anzahl theoretisch möglicher unterschiedlicher Basisfolgen bei n Bedingungen ist gleich der Anzahl der Permutationen der n Bedingungen.

Satz: Die Anzahl der Permutationen von n verschiedenen Elementen ist n![1)]

1.1.1.3.2 Bildung einer Basisfolge von Aktionen

Für die Bildung einer Basisfolge von Aktionen gilt:

Def.: Eine Basisfolge aus n Aktionen entsteht durch Anordnung dieser n Aktionen in einer Reihenfolge.

Zur Anzahl theoretisch möglicher Basisfolgen gilt das im Abschnitt 1.1.1.3.1 Ausgeführte analog.

1.1.1.3.3 Bildung von Bedingungsanzeigerkombinationen

Zunächst sollen elementare und quasi-elementare Bedingungsanzeigerkombinationen definiert werden.

Def.: Die durch das kartesische Produkt aller Bedingungsausprägungen gebildeten Bedingungsanzeigerkombinationen einer Basisfolge von Bedingungen werden als *elementare Bedingungsanzeigerkombinationen* bezeichnet.[2)]

---

1) Zu diesem Satz und seinem Beweis vgl. Mangoldt-Knopp (Höhere Mathematik), S. 8

2) Zur Ermittlung der Anzahl elementarer Bedingungsanzeigerkombinationen vgl. Montalbano (Decision tables), S. 66 ff.

Zu den beiden Bedingungen

Bedingung a mit [a] = {al,a2,a3}
Bedingung b mit [b] = {J,N}

lassen sich genau die folgenden elementaren Bedingungsanzeigerkombinationen bilden:

1	2	3	4	5	6
a1	a1	a2	a2	a3	a3
J	N	J	N	J	N

Elementare Bedingungsanzeigerkombinationen haben die Eigenschaft, sich nicht mehr in andere Bedingungsanzeigerkombinationen zerlegen zu lassen.

Def. : Die aus Bedingungsausprägungen und mindestens einem Ausschlußanzeiger bestehenden Bedingungsanzeigerkombinationen werden als *quasi-elementare Bedingungsanzeigerkombinationen* bezeichnet.

In dieser Bezeichnung kommt zum Ausdruck, dass sich diese Art von Bedingungsanzeigerkombinationen ebenfalls nicht mehr zerlegen lässt. "Quasi-elementar" soll jedoch darauf hinweisen, dass diese Bedingungsanzeigerkombinationen nicht alle Bedingungen der Basisfolge einschließen.

Wird nun noch der Indifferenzanzeiger zur Bildung von *Bedingungsanzeigerkombinationen* zugelassen, dann lässt sich die Menge der formal unterschiedlichen Bedingungsanzeigerkombinationen wie folgt bestimmen:

Def.: Die Menge K aller formal unterschiedlichen Bedingungsanzeigerkombinationen entspricht der Menge aller geordneten n-Tupel, die als kartesisches Produkt der jeweils um den Ausschlußanzeiger und den Indifferenzanzeiger erweiterten Menge der Bedingungsanzeiger aller n Bedingungen einer Basisfolge von Bedingungen entsteht:

$$K = ([a_i] \cup \{\sim\} \cup \{\#\}) \times ([b_j] \cup \{\sim\} \cup \{\#\}) \times \ldots \times ([Z_k] \cup \{\sim\} \cup \{\#\})$$

Zu den oben angegebenen beiden Bedingungen lassen sich dann genau 20 Bedingungsanzeigerkombinationen bilden.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
a1	a1	a1	a1	a2	a2	a2	a2	a3	a3	a3	a3	-	-	-	-	#	#	#	#
J	N	-	#	J	N	-	#	J	N	-	#	J	N	-	~	J	N	-	#

Eine Bedingungsanzeigerkombination, die nur aus Ausschlußanzeigern besteht, ist syntaktisch zulässig, jedoch semantisch bedeutungslos.

Ist in einer Bedingungsanzeigerkombination ein Indifferenzanzeiger enthalten, so steht diese Kombination für die Menge aller elementaren oder quasi-elementaren Kombinationen, die sich ergeben, wenn der Indifferenzanzeiger nacheinander durch alle Anzeiger der zugehörigen Bedingung ersetzt wird. Daraus ergibt sich, dass zwei Bedingungsanzeigerkombinationen im Verhältnis Exklusion, Inklusion und Überschneidung stehen können.

Bedingungsanzeigerkombinationen,. die zueinander in der Beziehung Exklusion stehen, werden als "disjunkt" bezeichnet. Bei Inklusion oder Überschneidung gelten sie als nicht disjunkt.

Nicht alle der formal zugelassenen Bedingungsanzeigerkombinationen sollen auch syntaktisch zulässig sein. Eine Einschränkung ergibt sich aus der Verwendung des Ausschlußanzeigers. Aus der Definition des Ausschlußanzeigers[1)] ergibt sich eine Vorschrift über die syntaktische Unzulässigkeit von Bedingungsanzeigerkombinationen:

Vorschrift: Wird zu einer Basisfolge von Bedingungen eine Bedingungsanzeigerkombination unter Benutzung eines Ausschlußanzeigers gebildet, dann sind alle weiteren Bedingungsanzeigerkombinationen syntaktisch unzulässig, bei denen in derselben Zeile kein Ausschlußanzeiger steht und für die gilt: Es gibt mindestens eine elementare oder quasi-elementare Kombination $k'_1 \in k_1$ und $k'_2 \in k_2$, die sich nur durch den Ausschlußanzeiger unterscheiden.

---

1) Vgl. Abschnitt 1.1.1.2.6

Beispiel:

Zu den Bedingungen

Bedingung a mit [a] = {al,a2,a3}
Bedingung b mit [b] = {J,N}
Bedingung c mit [c] = {c1,c2,c3}
Bedingung d mit [d] = {d1,d2}

wurde die Bedingungsanzeigerkombination

$$\begin{Bmatrix} a1 \\ \# \\ - \\ d1 \end{Bmatrix}$$

gebildet. Die folgenden Bedingungsanzeigerkombinationen sind Beispiele für nach dem obigen Satz unzulässige Kombinationen:

$$\begin{Bmatrix} a1 & a1 & a1 & \# \\ J & N & J & N \\ - & - & c1 & c2 \\ d1 & d1 & \# & d1 \end{Bmatrix}$$

Äußerlich unterschiedliche Mengen von Bedingungsanzeigerkombinationen können einander syntaktisch äquivalent sein.

Def.: Zwei Mengen von Bedingungsanzeigerkombinationen

$$\left\{k_1^r, k_2^r, \ldots k_n^r\right\} \quad \textit{und} \quad \left\{k_1^s, k_2^s, \ldots k_m^s\right\}$$

sind einander syntaktisch äquivalent, wenn

$$\bigcup_{v=1}^{n} k_v^r = k_\mu^S \cdot$$

Die syntaktische Äquivalenz wird erkennbar, wenn beide Mengen von Bedingungsanzeigerkombinationen in elementare Bedingungsanzeigerkombinationen transformiert werden. Die Mengen der Elementarkombinationen sind identisch.

Beispiel:

$$K^r = \left\{\begin{matrix} \underline{1} & \underline{2} & \underline{3} \\ J & J & N \\ J & N & N \\ - & - & N \\ N & N & N \\ - & - & N \end{matrix}\right\} \text{äqsyn} \quad K^S = \left\{\begin{matrix} \underline{1} & \underline{2} & \underline{3} & \underline{4} \\ J & J & J & - \\ - & J & N & N \\ - & - & J & N \\ N & N & N & N \\ J & N & N & N \end{matrix}\right\}$$

da

$$\nu \overset{n}{=} 1\, k_{\nu}^{r} = \mu \overset{m}{=} 1\, k_{\mu}^{S} =$$

1	2	3	4	5	6	7	8	9
J	J	J	J	J	J	J	J	N
J	J	J	J	N	N	N	N	N
J	J	N	N	J	J	N	N	N
N	N	N	N	N	N	N	N	N
J	N	J	N	J	N	J	N	N

1.1.1.3.4 Bildung von Aktionsanzeigerkombinationen

Wird bei der Bildung von Aktionsanzeigerkombinationen der Ausschlußanzeiger berücksichtigt, dann gilt:

Def.: Die Menge *K* aller formal unterschiedlichen *Aktionsanzeigerkombinationen* entspricht der Menge aller geordneten n-Tupel, die als kartesisches Produkt der jeweils um den Ausschlußanzeiger erweiterten Menge der Aktionsanzeiger aller n Aktionen einer Basisfolge von Aktionen entsteht. Alle Aktionsanzeigerkombinationen sind syntaktisch zulässig.

$$K = ([\alpha_i] \cup \{\sim\}) \times ([\beta_j] \cup \{\sim\}) \times \ldots \times ([\omega_k] \cup \{\sim\})$$

Beispiel: Zu den Aktionen
Aktion α mit $[\alpha] = \{\alpha 1, \alpha 2, \alpha 3, \alpha 4\}$
Aktion β mit $[\beta] = \{X\}$
Aktion γ mit $[\gamma] = \{\gamma 1, \gamma 2\}$

Lassen sich die folgenden 24 Aktionsanzeigerkombinationen bilden:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
α1	α1	α1	α1	α1	α1	α2	α2	α2	α2	α2	α2	α3	α3	α3	α3	α3	α3	~	~	~	~	~	~
X	X	X	~	~	~	X	X	X	~	~	~	X	X	X	~	~	~	X	X	X	~	~	~
γ1	γ2	~	γ1	γ2	~	γ1	γ2	~	γ1	γ2	~	γ1	γ2	~	γ1	γ2	~	γ1	γ2	~	γ1	γ2	~

Die formal gebildeten Anzeigerkombinationen stehen durch die Einbeziehung des Ausschlußanzeigers in den Beziehungen[1)] Exklusion, Inklusion und Überschneidung zueinander.

Satz: Zwei Aktionsanzeigerkombinationen

$\partial\ell_i \in K \quad und \quad \partial\ell_j \in K$

stehen

a) in der Beziehung *Exklusion* zueinander $(\partial\ell_i \cap \partial\ell_j = \phi)$, wenn in mindestens einer Zeile einander unterschiedliche Aktionsanzeiger gegenüberstehen, von denen keiner ein Ausschlußanzeiger sein darf,

b) in der Beziehung *Inklusion* zueinander $(\partial\ell_i \subset \partial\ell_j)$, wenn jeder Anzeiger von i mit seinem Gegenüber von j identisch ist oder ein Ausschlußanzeiger ist,

c) in der Beziehung *Überschneidung* zueinander $(\partial\ell_i \cap \partial\ell_j = \phi)$, wenn sie mindestens einen von einem Anschlussanzeiger verschiedenen Anzeiger gemeinsam haben und in einer weiteren Zeile unterschiedliche Anzeiger aufweisen.

1.1.1.3.5 Bildung von Entscheidungsregeln

Die Menge der syntaktisch zulässigen Entscheidungsregeln lässt sich wie folgt bestimmen.

Def.: Die Menge der syntaktisch zulässigen *Entscheidungsregeln* R entspricht der Menge aller geordneten Paare, die als kartesisches Produkt aus den syntaktisch zulässigen Bedingungs- und Aktionsanzeigerkombinationen der definierten Bedingungen und Aktionen gebildet werden können.

$$R = K \; x \; K$$

---

1) Bei der mengentheoretischen Betrachtung der Beziehungen von Aktionsanzeigerkombinationen sei auf folgenden Unterschied zur Betrachtung der Beziehungen von Bedingungsanzeigerkombinationen hingewiesen:
Während im letzteren Fall die elementare oder quasi-elementare Bedingungsanzeigerkombination als Element der Menge gilt, ist dies
bei einer Aktionsanzeigerkombination die durch einen Aktionsanzeiger symbolisierte einzelne Aktion.

Die in einer Entscheidungstabelle t enthaltenen Regeln R^t sind in der Regel eine Teilmenge dieser Produktmenge, denn die Menge der syntaktisch zulässigen Entscheidungsregeln enthält vielfach Anzeigerkombinationen, die semantisch bedeutungslos sind.

Für

Bedingung a mit [a] = {a1,a2,a3, a4}
Bedingung b mit [b] = {b1,b2,b3}
Bedingung c mit [c] = {J,N}

und

Aktion α mit [α] = {α1,α2,α3}
Aktion β mit [β] = {X}
Aktion γ mit [γ] = {γ1,γ2,γ3}

und eine Beschränkung auf elementare Bedingungsanzeigerkombinationen können 24 Bedingungsanzeigerkombinationen und 32 Aktionsanzeigerkombinationen gebildet werden. Es ergeben sich dadurch bereits 24 x 32 = 768 syntaktisch zulässige, unterschiedliche Entscheidungsregeln.

1.1.1.3.6 Bildung von Entscheidungstabellen

Die Bildung von Entscheidungstabellen lässt sich wie folgt definieren.

Def.: Jede *Entscheidungstabelle* entsteht durch Vereinigung einer definierten Basisfolge B von Bedingungen und einer Basisfolge A von Aktionen mit n zu diesen Bedingungen und Aktionen syntaktisch zulässigen Entscheidungsregeln R_v zu einem (n+1)-Tupel.

$$t = \left((B;A);R_{1;};R_2;...R_n\right) \quad R_v \subseteq R(v=1,...,n)$$

Unter den so entstehenden Entscheidungstabellen lassen sich zwei Arten unterscheiden:

Def.: a) Zur Bildung der *Eintreffer-Entscheidungstabelle* sind nur Regeln erlaubt, deren Bedingungsanzeigerkombinationen disjunkt sind.

b) Zur Bildung der *Mehrtreffer-Entscheidungstabelle* sind alle syntaktisch zulässigen Entscheidungsregeln erlaubt. [1)]

---

1) Zu dieser Definition wird ersichtlich, dass die Eintreffer-Entscheidungstabelle ein Sonderfall der Mehrtreffertabelle ist.

Für die elementaren syntaktischen Einheiten

Bedingung a mit [a] = {a1,a2,a3}
Bedingung b mit [b] = {b1,b2}
Bedingung c mit [c] = {J,N}

Aktion α mit [α] = {α1,α2,α3,α4}
Aktion β mit [β] = {X}
Aktion γ mit [γ] = {γ1,γ2}

wäre die folgende Entscheidungstabelle eine den angegebenen syntaktischen Bildungsregeln für die Eintreffertabelle entsprechende Tabelle:

Beispieltabelle		1	2	3	4	5	6
B1	Bedingung a	a1	a1	a2	a3	a3	a3
B2	Bedingung b	-	-	-	b1	b2	b2
B3	Bedingung c	J	N	-	-	J	N
A1	Aktion α	α1		α3	α3	α2	α4
A2	Aktion β		X	X	X	X	
A3	Aktion γ	γ1		γ2	γ2	γ1	γ2

Die folgende Beispieltabelle dagegen wäre als Eintreffer -Entscheidungstabelle syntaktisch nicht zulässig, da sich die Regeln 3 und 4 sowie die Regeln 5 und 6 im Bedingungsanzeigerteil überschneiden.

Beispieltabelle		1	2	3	4	5	6
B1	Bedingung a	a1	a1	a2	a2	a3	a3
B2	Bedingung b	-	-	-	b1	b2	b2
B3	Bedingung c	J	N	-	-	J	-
A1	Aktion α	α1		α3	α3	α2	α4
A2	Aktion β		X	X	X	X	
A3	Aktion γ	γ1		γ2	γ2	γ1	γ2

Diese Tabelle entspricht jedoch den Regeln einer Mehrtreffer -Entscheidungstabelle, die in einem eigenen Kapitel abgehandelt wird.

1.1.1.3.7 Bildung des Entscheidungstabellenverbunds

Die Ursache für eine Vereinigung von Entscheidungstabellen zu einem Entscheidungstabellenverbund kann nur im semantischen Bereich gesucht werden. Über seinen syntaktischen Aufbau lässt sich folgende Aussage machen:

Def.: Ein *Entscheidungstabellenverbund* entsteht durch Verknüpfung von Entscheidungstabellen.

Theoretisch lassen sich vier Verknüpfungsformen von Entscheidungsstabellen unterscheiden.

Bei der *Folge* werden zwei Entscheidungstabellen sequentiell hintereinander ausgeführt. Jede Regel der ersten beinhaltet als letzte Aktion einen Sprung zur zweiten Entscheidungstabelle.

Bei der *Verzweigung* kann von jeder Regel der ersten Tabelle zu einer anderen Tabelle verzweigt werden.

Bei der *Schleife* sind zunächst zwei Tabellen linear verknüpft.
Es wird jedoch zusätzlich eine Verknüpfung der zweiten Tabelle zurück zur ersten Tabelle hergestellt, so dass eine Schleife entsteht. Eine Regel der ersten oder eine Regel der zweiten Tabelle muss allerdings einen Aussprung aus der Schleife bewirken.

Bei der *Verschachtelung* wird die zweite Tabelle aus der Aktionsfolge der ersten Tabelle heraus aufgerufen und bearbeitet. Nach Bearbeitung der zweiten Tabelle erfolgt ein Rücksprung zur ersten Tabelle, bei der der Ablauf mit der nächsten Aktion fortgesetzt werden kann.

Bis zur Verbreitung des Konzepts der Strukturierten Programmierung herrschte in der Entscheidungstabellentechnik die Meinung, dass alle theoretisch möglichen Verknüpfungsformen zum Aufbau eines Entscheidungsstabellenverbunds angewendet werden können. Nunmehr erscheint es jedoch günstiger, in Analogie zur Verschachtelung der Elementarstrukturen der Strukturierten Programmierung, einen Entscheidungstabellenverbund nur noch durch verschachtelte Verknüpfung von Entscheidungstabellen aufzubauen.

Ein solchermaßen "strukturierter" Entscheidungstabellenverbund wird in der Regel einen hierarchischen Aufbau haben

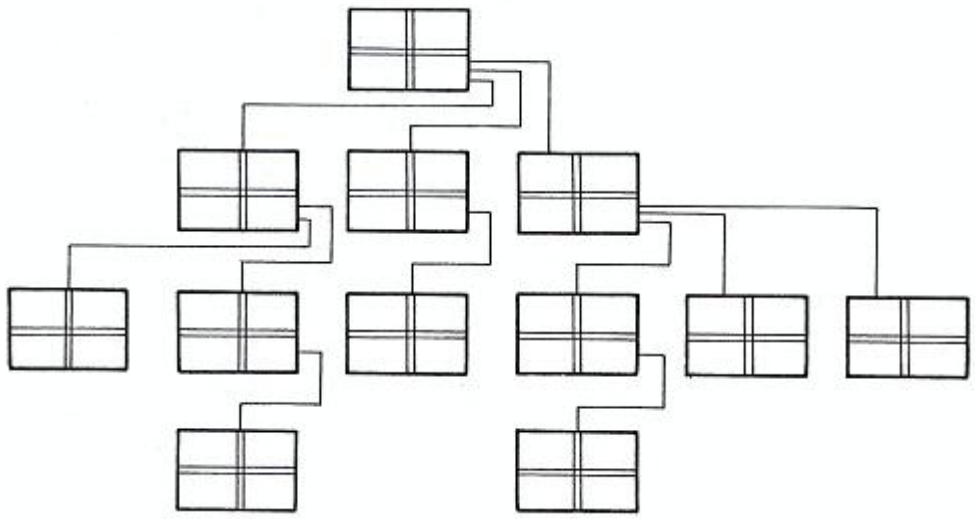


Abb. 2: Strukturierter Entscheidungsstabellenverbund

Es sind beliebig viele Verschachtelungsebenen möglich, so dass die syntaktischen Möglichkeiten der Konstruktion eines Entscheidungsstabellenverbunds jeder semantischen Anforderung gerecht
werden können.

1.1.1.4 Syntaktische Klassifikation von Entscheidungstabellen

Nach Syntaxmerkmalen lassen sich begrenzte, erweiterte und gemischte Entscheidungstabellen, vollständige und unvollständige Entscheidungsstabellen sowie Entscheidungstabellen in kanonischer Normalform unterscheiden.

1.1.1.4.1 Begrenzte, erweiterte und gemischte Entscheidungstabellen

Nach der Art der in einer Entscheidungstabelle benutzten Bedingungs- und Aktionsanzeiger lassen sich begrenzte, erweiterte und gemischte Entscheidungstabellen wie folgt definieren:

Def.: Entscheidungstabellen, die innerhalb der Entscheidungsregeln ausschließlich Bedingungsanzeigerkombinationen aus den Annzeigern "J", "N", "_" und "#" sowie Aktionsanzeigerkombinationen aus den Anzeigern "X" und "~" aufweisen, werden als *begrenzte Entscheidungstabellen*[1)] bezeichnet.

Begrenzte Entscheidungstabellen können folglich als elementare syntaktische Einheiten nur einfache Bedingungen und einfache Aktionen benutzen. Die Beispieltabelle ist nach dieser Definition eine begrenzte Entscheidungstabelle.

Beispieltabelle		1	2	3	4	5
B1	Bedingung a	J	J	J	N	N
B2	Bedingung b	J	J	N	J	N
B3	Bedingung c	J	N	-	-	N
A1	Aktion α			X		
A2	Aktion β	X		X	X	X
A3	Aktion γ		X			X

1) Im angelsächsischen Schrifttum überwiegend als "limited entry table" bezeichnet

Def.: Entscheidungstabellen, die innerhalb der Entscheidungsregeln weder die Bedingungsanzeiger "J" und "N" noch den Aktionsanzeiger "X" verwenden, werden als *erweiterte Entscheidungstabellen*[1)] bezeichnet.

Als elementare syntaktische Einheiten können demnach nur komplexe Bedingungen und Aktionen benutzt werden. Die folgende Beispieltabelle ist nach dieser Definition eine erweiterte Entscheidungstabelle.

Beispieltabelle		1	2	3	4
B1	Bedingung a	a1	a2	a2	a3
B2	Bedingung b	-	b1	b2	-
A1	Aktion α	α1		α1	α2
A2	Aktion β		β1	β1	β2

Def.: Entscheidungstabellen, in denen alle Anzeigertypen Verwendung finden, werden *gemischte Entscheidungstabellen*[2)] genannt.

In gemischten Entscheidungstabellen können sowohl einfache als auch komplexe Bedingungen und Aktionen benutzt werden. Die folgende Beispieltabelle ist demnach eine gemischte Entscheidungstabelle.

Beispieltabelle		1	2	3	4	5
B1	Bedingung a	J	J	N	N	N
B2	Bedingung b	b1	b2	-	-	-
B3	Bedingung c	-	-	c1	c2	c3
A1	Aktion α	X				X
A2	Aktion β		β2	β1		
A3	Aktion γ	γ3	γ1		γ1	γ2

1) Im angelsächsischen Schrifttum als "extended entry tables" bezeichnet
2) Im angelsächsischen Schrifttum als "mixed entry tables" bezeichnet

1.1.1.4.2 Vollständige und unvollständige Entscheidungstabellen

Vollständige und unvollständige Entscheidungstabellen lassen sich auf der Basis elementarer syntaktischer Einheiten und elementarer Bedingungsanzeigerkombinationen definieren.

Def.: Eine Entscheidungstabelle gilt als *syntaktisch vollständig*, wenn die in den Entscheidungsregeln aufgeführten Bedingungsanzeigerkombinationen alle formal zulässigen elementaren Bedingungsanzeigerkombinationen repräsentieren und jeder zulässige Aktionsanzeiger Bestandteil mindestens einer der aufgeführten Aktionsanzeigerkombinationen ist.

Elementare syntaktische Einheiten einer Entscheidungstabelle seien:

Bedingung a mit [a] = {a1,a2,a3,a4}
Bedingung b mit [b] = {b1,b2,b3}

Aktion α mit [α] = {α1,α2}
Aktion β mit [β] = {β1,β2,β3}

Eine syntaktisch zulässige Entscheidungstabelle auf der Basis dieser Elementareinheiten ist die folgende Beispieltabelle.

Beispieltabelle		1	2	3	4	5	6	7
B1	Bedingung a	a1	a2	a2	a2	a3	a4	a4
B2	Bedingung b	-	b1	b2	b3	-	b1	b2
A1	Aktion α	α1		α2	α2	α1	α2	α2
A2	Aktion β		β1	β2	β1	β2	β1	β2

Die Beispieltabelle ist jedoch syntaktisch unvollständig. Keine der sieben enthaltenen Regeln deckt die Bedingungsanzeigerkombination $\left\{\begin{matrix} a4 \\ b3 \end{matrix}\right\}$ ab. Auch der Aktionsanzeiger β3 fehlt in den aufgeführten Aktionsanzeigerkombinationen.

In jeder syntaktisch vollständigen Entscheidungstabelle lässt sich eine Gruppe von Regeln mit identischen Aktionsanzeigerkombinationen zu einer ELSE-Regel zusammenfassen.

Def.: Die *ELSE-Regel* repräsentiert in einer Entscheidungstabelle alle formal zulässigen elementaren Bedingungsanzeigerkombinationen, die nicht durch die anderen aufgeführten Entscheidungsregeln abgedeckt sind.[1)]

Beispieltabelle		1	2	3	4	5	ELSE
B1	Bedingung a	a1	a2	a2	a3	a4	
B2	Bedingung b	-	b1	b2	b1	-	
A1	Aktion α	α1		α2	α1		
A2	Aktion β		β1	β2	β3	β1	β2

Die ELSE-Regel in der Beispieltabelle repräsentiert die Bedingungs-Anzeigerkombination

$$\left\{\begin{matrix} a2 & a3 & a3 \\ b3 & b2 & b3 \end{matrix}\right\}$$

Es ist sinnvoll, innerhalb des Bedingungsanzeigerteils der ELSE-Regel alle Bedingungen ohne Anzeiger zu belassen, da ein direkter Einfluss der Bedingungen auf die Auslösung der beschriebenen Aktionsfolge nicht gegeben ist. Nur die anderen Entscheidungsregeln bestimmen den Inhalt des Bedingungsanzeigerteils der ELSE-Regel,

1.1.1.4.3 Entscheidungstabellen in kanonischer Normalform

Jede Entscheidungstabelle kann in ihre kanonische Normalform überführt werden.

Def.: Eine Entscheidungstabelle in *kanonischer Normalform*[2)] verfügt nur über Entscheidungsregeln mit elementaren Bedingungsanzeigerkombinationen. Die Regeln sind nach Maßgabe ihrer Bedingungsanzeigerkombinationen lexikographisch[3)] geordnet. Eine Entscheidungstabelle befindet sich in *vollständiger kanonischer Normalform*, wenn alle elementaren Bedingungsanzeigerkombinationen in den Regeln aufgeführt sind.

---

1) Die ELSE-Regel war als "´all other´ row" mit gleicher logischer Bedeutung bereits Bestandteil der ursprünglich bei General Electric entwickelten ersten Form der Entscheidungstabelle. Vgl. Kavanagh (TABSOL), S. 122

2) Der Ausdruck "kanonische Normalform" wurde gewählt, weil hier eine eindeutige Analogie zur kanonischen Normalform einer Aussagenfunktion gegeben ist. Die Bezeichnung "canonical form" wird ebenfalls angewendet bei Pollack (Analysis), S. 17. Er bezeichnet damit die nach Auflösung komplexer Regeln entstehenden elementaren Bedingungsanzeigerkombinationen.

3) Lexikographische Anordnung (Begriff aus der Kombinatorik) bedeutet, dass von zwei Regeln stets diejenige vorangeht, deren oberster Bedingungsanzeiger der niedrigere ist. Wenn die obersten Bedingungsanzeiger der beiden Regeln identisch sind, dann geht die voran, deren zweiter Bedingungsanzeiger der niedrigere ist, usw. Voraussetzung einer lexikographischen Anordnung ist also, dass die Bedingungsanzeiger jeder Bedingung in eine aufsteigende Ordnung gebracht werden.

Eine Entscheidungstabelle in vollständiger kanonischer Normalform[1)] verfügt also immer über das syntaktische Maximum an Entscheidungsregeln.

Für die elementaren syntaktischen Einheiten der im vorigen Abschnitt wiedergegebenen Beispieltabelle konnte folgende Entscheidungstabelle in vollständiger kanonischer Normalform erstellt werden:

Beispieltabelle		1	2	3	4	5	6	7	8	9	10	11	12
B1	Bedingung a	a1	a1	a1	a2	a2	a2	a3	a3	a3	a4	a4	a4
B1	Bedingung b	b1	b2	b3	b1	b2	b3	b1	b2	b3	b1	b2	b3
A1	Aktion α	α1	α1	α2	α1	α1	α1	α2	α2	α2	α1	α2	
A2	Aktion β		β3	β2	β1			β3	β3	β3	β1	β2	β3

Jede andere vollständige Entscheidungstabelle in kanonischer Normalform mit den gleichen Bedingungen und Aktionen als elementaren syntaktischen Einheiten kann sich höchstens im Aktionsanzeigerteil von der Beispieltabelle unterscheiden, denn es gilt folgender Satz:

Satz: Zu jeder vollständigen Entscheidungstabelle mit der Menge der Bedingungen B gibt es eine und nur eine Entscheidungstabelle in vollständiger kanonischer Normalform bezüglich B, die der ursprünglichen Entscheidungstabelle syntaktisch äquivalent ist.

Die Gültigkeit dieses Satzes[2)] ergibt sich schlüssig aus der Überlegung, dass jede Entscheidungsregel in syntaktisch äquivalente Regeln mit elementaren Bedingungsanzeigerkombinationen transformiert werden kann[3)] und Form und Anzahl aller elementaren Bedingungsanzeigerkombinationen durch die Bedingungen und ihre Anzeiger klar definiert sind.

Die kanonische Normalform der Entscheidungstabelle hat eine gewisse Bedeutung für die Tabellenerstellung. Es liegt nahe, zunächst den Bedingungsanzeigerteil einer Tabelle in vollständiger kanonischer Nor-

---

1) In der Literatur findet sich auch der Begriff "standard table" für eine begrenzte Entscheidungstabelle in vollständiger kanonischer Normal form. Vgl. Verhelst (Constructing decision tables), S. 28

2) Der Beweis des Satzes ließe sich auch in Analogie zum Beweis des Satzes über die Existenz und Eindeutigkeit einer kanonisch alternativen Normalform einer Aussagenfunktion führen. Vgl. dazu Asser (Mathematische Logik), S. 59 ff.

3) Vgl. dazu Abschnitt 1.1.2.1.2.1.1

malform aufzubereiten, um dann systematisch die Aktionsfolgen Regel für Regel zu ergänzen.[1)]

1.1.2 Transformationsregeln

Die syntaktische Dimension einer Sprache umfasst neben Regeln, nach denen syntaktische Einheiten gebildet werden können, auch Transformationsregeln, die die Möglichkeiten der syntaktischen Umwandlung dieser Einheiten definieren. Dabei lassen sich Transformationen syntaktischer Einheiten in Einheiten gleicher Stufe von Transformationen syntaktischer Einheiten in Einheiten niedrigerer oder höherer Stufe unterscheiden.

1.1.2.1 Transformationen syntaktischer Einheiten in syntaktische Einheiten gleicher Stufe

Sinnvolle Transformationen syntaktischer Einheiten in Einheiten gleicher Stufe sind bei Bedingungen, Aktionen, Bedingungsanzeigerkombinationen, Entscheidungsregeln und Entscheidungstabellen möglich.

1.1.2.1.1 Transformationen elementarer syntaktischer Einheiten

Als Transformationen elementarer syntaktischer Einheiten soll die Umwandlung komplexer Bedingungen bzw. Aktionen in einfache Bedingungen bzw. Aktionen erläutert werden.

1.1.2.1.1.1 Bedingungstransformation

Eine komplexe Bedingung lässt sich wie folgt in einfache Bedingungen transformieren:

Satz: Eine komplexe Bedingung mit n Anzeigern kann in eine Folge aus n einfachen Bedingungen transformiert werden. Zur Bildung jeder einfachen Bedingung wird der Basisteil der komplexen Bedingung mit einem Anzeiger vereinigt, und als neue Bedingungsanzeiger werden "J" oder "N" definiert.[2)]

---

1) Dieses "klassische" Tabellenerstellungsverfahren wird in Abschnitt 1.2.4.1 beschrieben.

2) Vgl. Pollack (Analysis), S. 6

Gegeben sei

Bedingung a mit [a] = { a1,a2,a3,a4 }

Syntaktisch äquivalent ist die Folge der folgenden 4 einfachen Bedingungen.

Bedingung a=a1 mit [a=a1] = {J,N}
Bedingung a=a2 mit [a=a2] = {J,N}
Bedingung a=a3 mit [a=a3] = {J,N}
Bedingung a=a4 mit [a=a4] = {J,N}

1.1.2.1.1.2 Aktionstransformation

Formal ist die Transformation einer komplexen Aktion in einfache Aktionen mit der Transformation einer komplexen Bedingung identisch.

Satz: Eine komplexe Aktion mit n Anzeigern kann in eine Folge von n einfachen Aktionen transformiert werden. Zur Bildung jeder einfachen Aktion wird der Basisteil der komplexen Aktion mit einem Anzeiger vereinigt, und als neuer Bedingungsanzeiger wird "X" definiert.

Danach lässt sich die komplexe Aktion
Aktion a mit [a] = {α1,α2,α3}

syntaktisch äquivalent in eine Folge von drei einfachen Aktionen transformieren:

Aktion $\alpha = \alpha 1$ mit $[\alpha = \alpha 1] = \{X\}$
Aktion $\alpha = \alpha 2$ mit $[\alpha = \alpha 2] = \{X\}$
Aktion $\alpha = \alpha 3$ mit $[\alpha = \alpha 3] = \{X\}$

1.1.2.1.2 Transformationen syntaktischer Einheiten höherer Stufe

Nicht alle syntaktischen Einheiten höherer Stufe können in syntaktisch äquivalente Alternativen transformiert werden. Von Bedeutung für die spätere Behandlung der semantischen und pragmatischen Dimension der Entscheidungstabelle sind die Transformationen von Entscheidungsregeln und Entscheidungstabellen.

1.1.2.1.2.1 Transformationen von Entscheidungsregeln

Die Möglichkeit, innerhalb der Bedingungsanzeigerkombination einer Entscheidungstabelle den Indifferenzanzeiger zu benutzen, schafft die Basis für Regeltransformationen. Die Auflösung einer komplexen Entscheidungsregel in einfache Entscheidungsregeln und die Zusammenfassung einfacher Entscheidungsregeln zu einer komplexen Entscheidungsregel sollen zunächst als "elementare" Regeltransformationen behandelt werden. Danach soll ein automatisierbares Verfahren zur Regelminimierung erläutert werden.

1.1.2.1.2.1.1 Elementare Regeltransformationen

Alle Regeltransformationen lassen sich auf zwei elementare Transformationen zurückführen: die Auflösung einer komplexen Regel und die Zusammenfassung einfacher Regeln.

Satz 1: Eine Entscheidungsregel mit mindestens einem Indifferenzanzeiger in der Bedingungsanzeigerkombination lässt sich in mehrere Entscheidungsregeln transformieren, deren Bedingungsanzeigerkombinationen aus den durch die ursprüngliche Bedingungsanzeigerkombination repräsentierten elementaren Bedingungsanzeigerkombinationen gebildet werden und deren Aktionsanzeigerkombinationen zur ursprünglichen identisch sind.[1)]

Beispiel:

Für

Bedingung a mit [a] = {a1,a2,a3}
Bedingung b mit [b] = {b1,b2}
Bedingung c mit [c] = {J,N}

und

Aktion α mit [α] = {α1,α2}
Aktion β mit [β] = {X}
Aktion γ mit [γ] = {γ1,γ2,γ3}

ist die Entscheidungsregel

$$\left\{\begin{array}{c} a1 \\ - \\ - \\ = \\ \alpha 2 \\ X \\ \gamma 3 \end{array}\right\}$$

syntaktisch zulässig.

1) Vgl. Montalbano (Decision tables), S. 55 und S. 63

Nach Satz 1 sind die folgenden 4 Entscheidungsregeln dazu syntaktisch äquivalent:

a1	a1	a1	a1
b1	b1	b2	b2
J	N	J	N
a2	a2	a2	a2
X	X	X	X
γ3	γ3	γ3	γ3

Die Anzahl der aus einer komplexen Regel entstehenden einfachen Regeln ergibt sich in Abhängigkeit von der Anzahl der Indifferenzanzeiger und der durch sie jeweils repräsentierten Bedingungsanzeiger. Eine komplexe Entscheidungsregel, deren Bedingungsanzeigerkombination n Indifferenzanzeiger enthält, die jeweils P_1, P_2, ... , P_n Bedingungsanzeiger repräsentieren, kann in

$$P_1 \,.\, P_2 \,.....\, P_n$$

einfache Regeln transformiert werden, die jeweils *elementare*[1)] Bedingungsanzeigerkombinationen enthalten.[2)]

Satz 2: Eine Gruppe von Entscheidungsregeln, die identische Aktionsanzeigerkombinationen enthalten, kann in eine einzige Entscheidungsregel transformiert werden, wenn es zu den aufgeführten Bedingungsanzeigerkombinationen eine einzige syntaktisch äquivalente Bedingungsanzeigerkombination gibt.[3)]

---

1) Enthält die komplexe Regel Ausschlußanzeiger, dann entstehen einfache Entscheidungsregeln mit quasi-elementaren Bedingungsanzeigerkombinationen.

2) Vgl. Montalbano (Tables, flow charts), S. 58

3) Für Bedingungen mit variabler Anzahl von Bedingungsanzeigern vgl. Montalbano (Decision tables), S. 54 f. und S. 61 f.

Beispiel: Für die in Erläuterung zu Satz 1 als Beispiel aufgeführten Bedingungen und Aktionen ist die folgende Gruppe von Entscheidungsregeln syntaktisch zulässig.

a1	a1	a2	a2	a3	a3
b1	b1	b1	b1	b1	b1
J	N	J	N	J	N
a2	a2	a2	a2	a2	a2
X	X	X	X	X	X
γ2	γ2	γ2	γ2	γ2	γ2

Nach Satz 2 für die Transformation von Entscheidungsregeln ist die folgende Entscheidungsregel syntaktisch äquivalent:

-
b1
-

α2
X
γ2

Zu einer Gruppe von Bedingungsanzeigern mit jeweils μ Bedingungsanzeigern gibt es dann eine einzige syntaktisch äquivalente Bedingungsanzeigerkombination mit λ Indifferenzanzeigern, wenn innerhalb der Gruppe die Bedingungsanzeiger in μ - λ Zeilen identisch sind und die Gruppe in den verbliebenen λ Zeilen alle formal zulässigen elementaren Bedingungsanzeigerkombinationen repräsentiert. Die entstehende Bedingungsanzeigerkombination enthält die Anzeiger, die in allen Bedingungsanzeigerkombinationen identisch waren, in gleicher Anordnung und an den übrigen Positionen Indifferenzanzeiger.

Beispiel:

Für

Bedingung a mit [a] = {a1,a2,a3}
Bedingung b mit [b] = {b1,b2}
Bedingung c mit [c] = {J,N}
Bedingung d mit [d] {dl ,d2 ,d3 ,d4}

sind folgende 7 Bedingungsanzeigerkombinationen zulässig:

$$\left\{\begin{array}{ccccccc} a1 & a1 & a2 & a3 & a3 & a3 & a3 \\ b1 & b2 & - & - & - & - & - \\ J & J & J & J & J & J & J \\ - & - & - & d1 & d2 & d3 & d4 \end{array}\right\}$$

Dazu ist die Bedingungsanzeigerkombination

$$\left\{\begin{array}{c} - \\ - \\ J \\ - \end{array}\right\}$$

syntaktisch äquivalent, da der Anzeiger "J" in allen Kombinationen erscheint und die verbleibenden Anzeiger mit

$$\left\{\begin{array}{ccccccc} a1 & a1 & a2 & a3 & a3 & a3 & a3 \\ b1 & b2 & - & - & - & - & - \\ - & - & - & d1 & d2 & d3 & d4 \end{array}\right\}$$

alle formal möglichen elementaren Kombinationen repräsentieren.

1.1.2.1.2.1.2 Verfahren zur Regelminimierung

Verfügt eine Gruppe von Regeln innerhalb einer Entscheidungstabelle über identische Aktionsanzeigerkombinationen, dann ist eine Analyse zur Ermittlung ihrer minimalen Repräsentation von erheblicher praktischer Bedeutung. Eine Entscheidungstabelle gilt in Bezug auf ihren Regelteil als minimiert, wenn alle Gruppen von Regeln mit jeweils identischen Aktionsanzeigerkombinationen in ihre minimale Repräsentation transformiert worden sind. Üblicherweise wird empfohlen, dies stufenweise zu tun, das heißt, immer nur Teilgruppen von Regeln so zusammenzufassen, dass jeweils eine Bedingung als indifferent ausgewiesen werden kann.[1)]

Das im Folgenden beschriebene Verfahren betrachtet die Gruppe der Regeln mit identischen Aktionsanzeigerkombinationen als Gesamtheit. Es besteht im Wesentlichen aus Vergleichsoperationen und würde sich leicht automatisieren lassen.

---

1) Vgl. Pollack; Hicks; Harrison (Decision tables), S. 56 ff; Montalbano (Decision tables), S. 54 f; und S. 61 f.

Unter folgenden Prämissen wird das Verfahren erst durchführbar:

- alle Bedingungsanzeiger einer Bedingung müssen explizit formuliert sein, entweder innerhalb der Regeln oder in einer separaten Definition,
- die vorhandenen Bedingungsanzeigerkombinationen sind zueinander disjunkt.

Für jede Gruppe von Regeln mit identischen Aktionsanzeigerkombinationen wird das Verfahren in folgenden Teilschritten ausgeführt:

1. Prüfen, in welchen Bedingungszeilen alle Bedingungsanzeiger der betreffenden Bedingung mindestens je einmal auftreten. Solche Bedingungszeilen besonders kennzeichnen.

2. Minimierungs-Analyse auf der Basis der obersten gekennzeichneten Bedingungszeile in folgenden Stufen:

   - Regeln, die in dieser Zeile einen Indifferenzanzeiger aufweisen, bleiben unberücksichtigt,

   - die verbliebenen Regeln werden untereinander verglichen, wobei jedoch die Bedingungsanzeiger in der untersuchten Zeile unberücksichtigt bleiben. Wenn die Bedingung der untersuchten Zeile i Anzeiger hat, dann werden alle Gruppen aus identischen Regeln identifiziert.

   - Jede identifizierte Regelgruppe wird zu einer einzigen komplexen Regel zusammengefasst, wobei die Bedingung der untersuchten Bedingungszeile einen Indifferenzanzeiger erhält.

   - Nach der Zusammenfassung der Regelgruppen prüfen, ob immer noch alle Bedingungsanzeiger in der untersuchten Zeile explizit vorhanden sind. Wenn das der Fall ist, bleibt die Bedingungszeile gekennzeichnet, andernfalls wird das Kennzeichen gelöscht.

3. Alle anderen gekennzeichneten Bedingungszeilen nach der unter 2. beschriebenen Prozedur behandeln.

4. Beendigung der Minimierungsanalyse, wenn

   a) keine Bedingungszeile mehr gekennzeichnet ist oder

   b) nach dem letzten erfolgreichen Minimierungsschritt alle noch gekennzeichneten Bedingungszeilen erfolglos überprüft worden sind.

Beispiel:

Prämissen:

Bedingung a mit [a] = {al,a2,a3}
Bedingung b mit [b] = {bl,b2,b3}
Bedingung e mit [e] = {el,e2}
Bedingung d mit [d] = {dl,d2,d3}

		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
B1	Bedingung a	-	a2	a2	a2	a1	a1	a1	a2	a2	a2	a1	a3	a2	a3	a3	a3
B2	Bedingung b	b1	b2	b2	b2	b3	b3	b3	b3	b3	b3	b1	b3	b1	b3	b3	b1
B3	Bedingung c	c1	c2	c2	c2	-	-	-	-	-	-	c2	-	c2	-	-	c2
B4	Bedingung d	-	d2	d1	d3	d1	d2	d3	d3	d2	d1	-	d1	-	d2	d3	-
A1	Aktion α	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

1. Schritt: Explizite Präsenz der Bedingungsanzeiger
   In jeder Zeile können jeweils alle Bedingungsanzeiger ermittelt werden. Alle Bedingungszeilen werden gekennzeichnet.

2. Schritt: Minimierungsanalyse auf der Basis der Bedingungszeile B1
   Regel 1 bleibt unberücksichtigt, da sie in der B1-Zeile einen Indifferenzanzeiger aufweist.
   Alle anderen Regeln werden untereinander verglichen, wobei Regelgruppen zu identifizieren sind, die aus jeweils 3 in den B2-, B3- und B4-Zeilen identischen Regeln bestehen.

   Es werden folgende Gruppen identifiziert:

   1) R5, R10, R12
   2) R6, R9, R14
   3) R7, R8, R15
   4) R11, R13, R16

   Die Regeln R5, R6, R7 und R11 erhalten einen Indifferenzanzeiger in der B1-Zeile, alle anderen Regeln werden gelöscht.

Danach verbleiben noch folgende Regeln:

		1	2	3	4	5	6	7	8
B1	Bedingung a	-	a2	a2	a2	-	-	-	-
B2	Bedingung b	b1	b2	b2	b2	b3	b3	b3	b1
B3	Bedingung c	c1	c2	c2	c2	-	-	-	c2
B4	Bedingung d	-	d2	d1	d3	d1	d2	d3	-
A1	Aktion α	X	X	X	X	X	X	X	X

Da in der B1-Zeile nicht mehr alle Bedingungsanzeiger explizit erscheinen, wird die Kennzeichnung gelöscht.

3. Schritt:

a) Minimierungsanalyse auf der Basis der Bedingungszeile B2

Vergleich der Regeln untereinander in den Zeilen Bl, B3 und B4. Gruppen aus jeweils 3 identischen Regeln lassen sich nicht ermitteln, so dass keine Regeln zusammengefasst werden können.

Die Bedingungszeile B2 bleibt jedoch gekennzeichnet

b) Minimierungsanalyse auf der Basis der Bedingungszeile B3

Die Regeln 5, 6 und 7 bleiben wegen ihres Indifferenzanzeigers in der B3-Zeile unberücksichtigt.

Alle anderen Regeln werden in den Bl-, B2- und B4-Zeilen auf Identität überprüft. Gruppen aus jeweils 2 identischen Regeln sind zu identifizieren. Es wird eine Gruppe, bestehend aus Rl und R8, ermittelt.

Rl erhält einen Indifferenzanzeiger in der B3-Zeile, R8 wird gelöscht.

Danach verbleiben noch folgende Regeln:

		1	2	3	4	5	6	7
B1	Bedingung a	-	a2	a2	a2	-	-	-
B2	Bedingung b	b1	b2	b2	b2	b3	b3	b3
B3	Bedingung c	-	c2	c2	c2	-	-	-
B4	Bedingung d	-	d2	d1	d3	d1	d2	d3
A1	Aktion α	X	X	X	X	X	X	X

Die Kennzeichnung der B3-Zeile wird gelöscht.

c) Minimierungsanalyse auf der Basis der Bedingungszeile B4

Regel 1 bleibt wegen ihres Indifferenzanzeigers in der B4-Zeile unberücksichtigt.

Alle anderen Regeln werden in den B1-, B2- und B3-Zeilen auf Identität überprüft. Gruppen aus jeweils 3 identischen Regeln sind zu identifizieren.

1) R2, R3, R4

2) R5, R6, R7

R2 und R5 erhalten einen Indifferenzanzeiger in der B4-Zeile, die übrigen Regeln der Gruppen werden gelöscht.

Danach verbleiben noch folgende Regeln:

	1	2	3
B1	-	a2	-
B2	b1	b2	b3
B3	-	c2	-
B4	-	-	-
A1	X	X	X

Die Kennzeichnung der B4-Zeile wird gelöscht.

4. Schritt: Nur die B2-Zeile ist noch gekennzeichnet. Ein erneuter Minimierungsversuch verläuft ergebnislos. Die Minimierung ist beendet.

Da bei der Durchführung des Verfahrens bestehende komplexe Regeln unverändert in die Vergleichsoperationen einbezogen werden, kann die minimale Regelanzahl als Ergebnis nicht garantiert werden. Ist beispielsweise eine elementare Bedingungsanzeigerkombination bereits durch eine komplexe Entscheidungsregel repräsentiert, kann sie nicht mehr zur Minimierung einer anderen Teilgruppe der Regeln herangezogen werden, auch wenn der Minimierungseffekt dann größer wäre. Dieser Schwäche des Verfahrens kann dadurch begegnet werden, dass zunächst alle vorhandenen komplexen Regeln in einfache Regeln aufgelöst werden.

Es besteht dann jedoch immer noch die Gefahr, dass bei mechanischer Abarbeitung der Entscheidungstabelle von oben nach unten Bedingungsanzeigerkombinationen zu früh in komplexe Ausdrücke eingehen und dass sich bei ihrer Zusammenfassung mit anderen Kombinationen insgesamt günstigere Ergebnisse erzielen ließen. Dieser Schwäche des Verfahrens kann durch folgende Strategie begegnet werden:

Die Minimierungsanalyse wird für alle Bedingungszeilen ausgeführt, ohne dass eine Transformation von Regelgruppen in komplexe Regeln tatsächlich erfolgt. Wenn feststeht, bei welcher Bedingungszeile sich die größte Anzahl zusammenfassbarer Gruppen ergibt, dann wird bei dieser Zeile transformiert. Wird bei mehreren Zeilen eine identische Anzahl zusammenfassbarer Gruppen ermittelt, dann wird die Zeile der Bedingung mit der größten Anzahl an Bedingungsanzeigern ausgewählt. Wenn auch dann noch mehr als eine Zeile zur Auswahl steht, kann eine dieser Zeilen beliebig bestimmt werden.

Durch diese Strategie wird zwar die Anzahl der notwendigen Vergleichsoperationen vervielfacht, bei maschineller Ausführung fällt jedoch dieser zusätzliche Aufwand kaum ins Gewicht.

1.1.2.1.2.2 Transformationen von Entscheidungstabellen

Als bedeutendste syntaktische Transformationen von Entscheidungstabellen sollen im Folgenden die Auflösung erweiterter oder gemischter Entscheidungstabellen in begrenzte Entscheidungstabellen und die Umwandlung von Entscheidungstabellen in ihre kanonische Normalform behandelt werden.

1.1.2.1.2.2.1 Auflösung erweiterter und gemischter Entscheidungstabellen in begrenzte Entscheidungstabellen

Begrenzte Entscheidungstabellen enthalten als elementare syntaktische Einheiten nur einfache Bedingungen und Aktionen.[1)] Die Transformation einer erweiterten oder gemischten Entscheidungstabelle in eine begrenzte Entscheidungstabelle kann in folgenden Schritten erfolgen:[2)]

1. Alle enthaltenen komplexen Bedingungen und Aktionen werden in einfache Bedingungen und Aktionen aufgelöst.

2. Alle Entscheidungsregeln werden in syntaktisch äquivalente Entscheidungsregeln umgewandelt, die nur noch aus den bei einfachen Bedingungen und Aktionen, zugelassenen Anzeigern "J", "N", "-", "#", "X" und "~" (Leerstelle) bestehen.

   Diese Umwandlung erfolgt nach folgenden Vorschriften:

   a) Ist eine komplexe Bedingung in einer Regel einer erweiterten oder gemischten Entscheidungstabelle als indifferent ausgewiesen, so werden alle aus dieser komplexen Bedingung entstehenden einfachen Bedingungen in der transformierten Regel als indifferent ausgewiesen.

   b) Wird eine komplexe Bedingung durch "#" oder eine komplexe Aktion durch "~" (Leerstelle) innerhalb der Regel einer erweiterten oder gemischten Entscheidungstabelle ausgeschlossen, so werden innerhalb der transformierten Regeln alle aus dieser komplexen Bedingung oder Aktion entstehenden einfachen Bedingungen oder Aktionen durch "#" bzw. "~" ausgeschlossen.

   c) Jeder Anzeiger y_i jeder komplexen Bedingung y wird innerhalb einer Entscheidungsregel in den Anzeiger "J" bei der aus der Bedingung y mit $[y] = y_i$ gebildeten einfachen Bedingung transformiert. Alle anderen aus der komplexen Bedingung gebildeten einfachen Bedingungen erhalten den Bedingungsanzeiger "N".

   d) Jeder Anzeiger pj jeder komplexen Aktion p wird innerhalb einer Entscheidungsregel in den Anzeiger "X" bei der aus der Aktion p mit [p] = pj gebildeten einfachen Aktion transformiert. Alle anderen aus der komplexen Aktion gebildeten einfachen Aktionen werden durch den Anzeiger "~" von der Ausführung innerhalb der transformierten Regel ausgeschlossen.

---

1) Vgl. Abschnitt 1.1.1.4.1

2) Zur Transformation erweiterter oder gemischter Entscheidungstabellen in begrenzte Entscheidungstabellen vgl. auch Press (Conversion of decision tables), S. 388 f .. Das von Press vorgeschlagene Verfahren führt jedoch zu mehrdeutigen Entscheidungstabellen.

Beispiel:

Die folgende gemischte Entscheidungstabelle soll in eine syntaktisch äquivalente begrenzte Entscheidungstabelle transformiert werden.

		1	2	3	4	5	6
B1	Bedingung a	a1	a1	a1	a2	a3	a3
B2	Bedingung b	b1	b2	b2	-	b1	b2
B3	Bedingung c	-	J	N	-	-	-
A1	Aktion α	$\alpha 2$			$\alpha 1$	$\alpha 2$	$\alpha 2$
A2	Aktion β		X	X		X	
A3	Aktion γ	$\gamma 2$		$\gamma 1$		$\gamma 1$	$\gamma 3$

Schritt 1 ergibt:

B1	Bedingung a = a1 mit [a = a1] = {J,N}
B2	Bedingung a = a2 mit [a = a2] = {J,N}
B3	Bedingung a = a3 mit [a = a3] = {J,N}
B4	Bedingung b = b1 mit [b = b1] = {J,N}
B5	Bedingung b = b2 mit [b = b2] = {J,N}
B6	Bedingung c mit [c] = {J,N}
A1	Aktion $\alpha = \alpha 1$ mit $[\alpha = \alpha 1] = \{\chi\}$
A2	Aktion $\alpha = \alpha 2$ mit $[\alpha = \alpha 2] = \{\chi\}$
A3	Aktion β mit $[\beta] = \{\chi\}$
A4	Aktion $\gamma = \gamma 1$ mit $[\gamma = \gamma 1] = \{\chi\}$
A5	Aktion $\gamma = \gamma 2$ mit $[\gamma = \gamma 2] = \{\chi\}$
A6	Aktion $\gamma = \gamma 3$ mit $[\gamma = \gamma 3] = \{\chi\}$

Schritt 2 ergibt:

		1	2	3	4	5	6
B1	Bedingung a = a1	J	J	J	N	N	N
B2	Bedingung b = a2	N	N	N	J	N	N
B3	Bedingung c = a3	N	N	N	N	J	J
B4	Bedingung b = b1	J	N	N	--	J	N
B5	Bedingung b = b2	N	J	J	--	N	J
B6	Bedingung c	--	J	N	--	--	--
A1	Aktion $\alpha = \alpha 1$				X		
A2	Aktion $\alpha = \alpha 2$	X				X	X
A3	Aktion β		X	X		X	
A4	Aktion $\gamma = \gamma 1$			X		X	
A5	Aktion $\gamma = \gamma 2$	X					
A6	Aktion $\gamma = \gamma 3$						X

1.1.2.1.2.2.2 Umwandlung von Entscheidungstabellen in ihre kanonische Normalform

Ein einfaches Umwandlungsverfahren besteht darin, alle in einer Tabelle enthaltenen komplexen Entscheidungsregeln in einfache Entscheidungsregeln aufzulösen und diese zusammen mit den ursprünglich vorhandenen einfachen Regeln lexikographisch anzuordnen. Um den bei größeren Tabellen mit der lexikographischen Anordnung der Regeln verbundenen Aufwand einzusparen, kann auch folgendes Verfahren angewendet werden:[1)]

1. Errechnen der Anzahl elementarer Bedingungsanzeigerkombinationen K_{max} durch Multiplikation der Bedingungsanzeigeranzahl A_i jeder Bedingung i miteinander,

2. Errechnen des Wiederholungsfaktors W_i zur systematischen Eintragung der Bedingungsanzeiger der Bedingung i in die Standardtabelle durch die Vorschrift

$$W_i = \frac{W_{i-1}}{A_i} .$$

Dabei gilt für i = 1 die Identität $W_o = K_{max}$.

1) Vgl. London (Decision Tables), S. 48 ff.

3. Zeilenweises Eintragen der Bedingungsanzeiger nach Maßgabe der errechneten Wiederholungsfaktoren.

4. Übernahme der Aktionsanzeigerkombinationen der ursprünglichen Regeln in alle Regeln der Standardtabelle, deren Bedingungsanzeigerkombination im Vergleich zur Bedingungsanzeigerkombination der ursprünglichen Regel identisch ist oder eine Teilmenge darstellt.

Beispiel:

Für

Bedingung	a	mit	[a]	=	{al,a2,a3,a4}
Bedingung	b	mit	[b]	=	{J,N}
Bedingung	c	mit	[c]	=	{cl,c2,c3}

und

Aktion α	mit	[α]	=	{α1,α2,α3}
Aktion β	mit	[β]	=	{X}
Aktion γ	mit	[γ]	=	{γ1,γ2}

ist folgende Entscheidungstabelle erstellt worden:

		1	2	3	4	5	6	7
B1	Bedingung a	a1	a1	a2	a3	a3	a3	a4
B2	Bedingung b	J	N	-	-	-	-	-
B3	Bedingung c	-	-	-	c1	c2	c3	-
A1	Aktion α		α2	α3		α1	α3	
A2	Aktion β	X			X	X		X
A3	Aktion γ	γ1		γ2			γ3	

Sie soll in eine syntaktisch äquivalente Standardtabelle transformiert werden.

1.Schritt: Anzahl elementarer Bedingungsanzeigerkombinationen

K_{max} = 4 * 2 * 3 = 24

2.Schritt: Wiederholungsfaktoren

Bedingung A: $\frac{K_{max}}{4} = \frac{24}{4} = \underline{\underline{6}}$

Bedingung B: $\frac{6}{2} = \underline{\underline{3}}$

Bedingung C: $\frac{3}{3} = \underline{\underline{1}}$

3. Eintragen der Bedingungsanzeiger unter Beachtung der errechneten Wiederholungsfaktoren

			1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
B1	Bedingung a		a1	a1	a1	a1	a1	a1	a2	a2	a2	a2	a2	a2	a3	a3	a3	a3	a3	a3	a4	a4	a4	a4	a4	a4
B2	Bedingung b		J	J	J	N	N	N	J	J	J	N	N	N	J	J	J	N	N	N	J	J	J	N	N	N
B3	Bedingung c		c1	c2	c3	c1	c2	c3	c1	c2	c3	c1	c2	c3	c1	c2	c3	c1	c2	c3	c1	c2	c3	c1	c2	c3
A1	Aktion α																									
A2	Aktion β																									
A3	Aktion γ																									

4. Übernahme der Aktionsanzeigerkombinationen

			1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
B1	Bedingung a		a1	a1	a1	a1	a1	a1	a2	a2	a2	a2	a2	a2	a3	a3	a3	a3	a3	a3	a4	a4	a4	a4	a4	a4
B2	Bedingung b		J	J	J	N	N	N	J	J	J	N	N	N	J	J	J	N	N	N	J	J	J	N	N	N
B3	Bedingung c		c1	c2	c3	c1	c2	c3	c1	c2	c3	c1	c2	c3	c1	c2	c3	c1	c2	c3	c1	c2	c3	c1	c2	c3
A1	Aktion α					α2	α2	α2	α3	α3	α3	α3	α3	α3		α1	α3		α1	α3						
A2	Aktion β		X	X	X										X	X		X	X		X	X	X	X	X	X
A3	Aktion γ		γ1	γ1	γ1					γ2	γ2	γ2	γ2	γ2			γ1			γ1						

1.1.2.2 Transformationen syntaktischer Einheiten in syntaktische Einheiten anderer Stufe

Relevante Transformationen syntaktischer Einheiten in syntaktische Einheiten anderer (höherer oder niedrigerer) Stufe sind die Zergliederung von Entscheidungstabellen zu einem Entscheidungstabellenverbund und die Kombination von Entscheidungstabellen.

1.1.2.2.1 Zergliederung von Entscheidungstabellen

Die "Zergliederung" von Entscheidungstabellen lässt sich formal als syntaktische Transformation einer Entscheidungstabelle in einen Entscheidungstabellenverbund begreifen. Es werden Transformationsregeln benötigt, nach denen die Aufteilung der Bedingungen und Aktionen der monolithischen Entscheidungstabelle auf die entstehenden Teiltabellen erfolgen kann und außerdem die Art der Verknüpfung der unterschiedlichen Teiltabellen untereinander definiert wird.

Einfache syntaktische Zergliederungsverfahren sind die Zergliederung durch *Regelaufteilung* und die Zergliederung durch *Bedingungsaufteilung*.

Bei der Zergliederung durch *Regelaufteilung* [1)] werden zwei oder mehr Teiltabellen gebildet, die linear angeordnet werden. Die Bedingungen und Aktionen der Teiltabellen entsprechen der ursprünglichen monolithischen Tabelle, die Regeln der ursprünglichen Tabelle sind jedoch auf die Teiltabellen aufteilt.

Beispiel zur Zergliederung durch Regelaufteilung:

TAB 1			1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
B1	Bedingung a		a1	a1	a1	a1	a1	a1	a2	a2	a2	a2	a2	a2	a3	a3	a3	a3	a3	a3	a4	a4	a4	a4	a4	a4
B2	Bedingung b		b1	b1	b1	b2	b2	b2	b1	b1	b1	b2	b2	b2	b1	b1	b1	b2	b2	b2	b1	b1	b1	b2	b2	b2
B3	Bedingung c		c1	c2	c3	c1	c2	c3	c1	c2	c3	c1	c2	c3	c1	c2	c3	c1	c2	c3	c1	c2	c3	c1	c2	c3
A1	Aktion α		α3	α1	α2	α1			α3	α1	α2	α1			α3	α1	α2	α1			α3	α1	α2	α1		
A2	Aktion β		β3	β2	β2			β1	β3	β2	β2			β1	β3	β2	β2			β1	β3	β2	β2			β1
A3	Aktion γ		γ1	γ1	γ1	γ1	γ1	γ1	γ2	γ2	γ2	γ2	γ2	γ2	γ3	γ3	γ3	γ3	γ3	γ3						

1) Vgl. dazu Lauer (Auflösung von Entscheidungstabellen), S.438

Die Anzahl der Teiltabellen kann frei gewählt werden. Wenn die ursprüngliche Tabelle in zwei Teiltabellen aufgelöst werden soll, dann ergibt sich folgender Entscheidungstabellenverbund:

TAB 1.1			1	2	3	4	5	6	7	8	9	10	11	12	Else
B1	Bedingung a		a1	a1	a1	a1	a1	a1	a2	a2	a2	a2	a2	a2	
B2	Bedingung b		b1	b1	b1	b2	b2	b2	b1	b1	b1	b2	b2	b2	
B3	Bedingung c		c1	c2	c3	c1	c2	c3	c1	c2	c3	c1	c2	c3	
A1	Aktion α		α3	α1	α2	α1			α3	α1	α2	α1			
A2	Aktion β		β3	β2	β2			β1	β3	β2	β2			β1	
A3	Aktion γ		γ1	γ1	γ1	γ1	γ1	γ1	γ2	γ2	γ2	γ2	γ2	γ2	
A4	Gehe nach TAB 1.2														X

TAB 1.2			1	2	3	4	5	6	7	8	9	10	11	12
B1	Bedingung a		a3	a3	a3	a3	a3	a3	a4	a4	a4	a4	a4	a4
B2	Bedingung b		b1	b1	b1	b2	b2	b2	b1	b1	b1	b2	b2	b2
B3	Bedingung c		c1	c2	c3	c1	c2	c3	c1	c2	c3	c1	c2	c3
A1	Aktion α		α3	α1	α2	α1			α3	α1	α2	α1		
A2	Aktion β		β3	β2	β2			β1	β3	β2	β2			β1
A3	Aktion γ		γ3	γ3	γ3	γ3	γ3	γ3						

Bei der Zergliederung durch *Bedingungsaufteilung*[1)] werden die Bedingungen der ursprünglichen Tabelle in mehrere Gruppen zerlegt. Die Anzahl der resultierenden Teiltabellen kann nicht frei gewählt werden. Sie ergibt sich in Abhängigkeit von der Anzahl der Bedingungsanzeiger der ausgegliederten Bedingungen. Die entstehenden Teiltabellen bilden im Entscheidungstabellenverbund eine Verzweigungsanordnung.

---

1) Sogenannte "Entscheidungsbaum"-Algorithmen zergliedern eine Entscheidungstabelle durch konsequente Bedingungsaufteilung vollständig bis in Einheiten mit jeweils einer Bedingungsabfrage. Diese Algorithmen basieren auf den Entwicklungen von Montalbano und Pollack. Vgl. Montalbano (Tables, flow charts), S. 54 ff. und Pollack (Conversion of limited-entry decision tables), S. 678 ff. Bei der "Zergliederung durch Bedingungsaufteilung" handelt es sich um nichts anderes als um eine Anwendung dieses Algorithmus, allerdings nicht mit letzter Konsequenz, sondern nur bis zu einer Stufe, in der Teiltabellen mit akzeptabler Größe entstehen.

Folgender Entscheidungstabellenverbund entsteht, wenn in der Beispieltabelle die erste Bedingung ausgegliedert wird:

TAB 1.0			1	2	3	4
B1	Bedingung a		a1	a2	a3	a4
A1	Gehe nach TAB		1.1.	1.2.	1.3.	1.3.

TAB 1.1			1	2	3	4	5	6
B1	Bedingung b		b1	b1	b1	b2	b2	b2
B2	Bedingung c		c1	c2	c3	c1	c2	c3
A1	Aktion α		α3	α1	α2	α1		
A2	Aktion β		β3	β2	β2			β1
A3	Aktion γ		γ1	γ1	γ1	γ1	γ1	γ1

TAB 1.2			1	2	3	4	5	6
B1	Bedingung b		b1	b1	b1	b2	b2	b2
B2	Bedingung c		c1	c2	c3	c1	c2	c3
A1	Aktion α		α3	α1	α2	α1		
A2	Aktion β		β3	β2	β2			β1
A3	Aktion γ		γ2	γ2	γ2	γ2	γ2	γ2

TAB 1.3			1	2	3	4	5	6
B1	Bedingung b		b1	b1	b1	b2	b2	b2
B2	Bedingung c		c1	c2	c3	c1	c2	c3
A1	Aktion α		α3	α1	α2	α1		
A2	Aktion β		β3	β2	β2			β1
A3	Aktion γ		γ3	γ3	γ3	γ3	γ3	γ3

TAB 1.4			1	2	3	4	5	6
B1	Bedingung b		b1	b1	b1	b2	b2	b2
B2	Bedingung c		c1	c2	c3	c1	c2	c3
A1	Aktion α		α3	α1	α2	α1		
A2	Aktion β		β3	β2	β2			β1
A3	Aktion γ							

Ein anspruchsvolleres Verfahren ist die Zergliederung durch *kombinierte Bedingungs-/Aktionsaufteilung*.[1)] Die Beziehungen der Bedingungen zu den Aktionen werden untersucht und solche Gruppen von Bedingungen und Aktionen ausgegliedert, die zu den verbliebenen Bedingungen und Aktionen keine logischen Beziehungen aufweisen.

Die logischen Beziehungen einer Bedingung zu den Aktionen oder vice versa sind aus der Entscheidungstabelle nicht ablesbar. Da innerhalb jeder Regel eine Aktionsanzeigerkombination einer Bedingungsanzeigerkombination zugeordnet ist, kann der Einfluss einer einzigen Bedingung auf die Aktionen nicht ohne zusätzliche Analyse festgestellt werden.

Diese Analyse wird in folgenden Schritten vorgenommen:

1. Pro Aktion Zusammenstellen aller Bedingungsanzeigerkombinationen, denen diese Aktion zugeordnet ist.

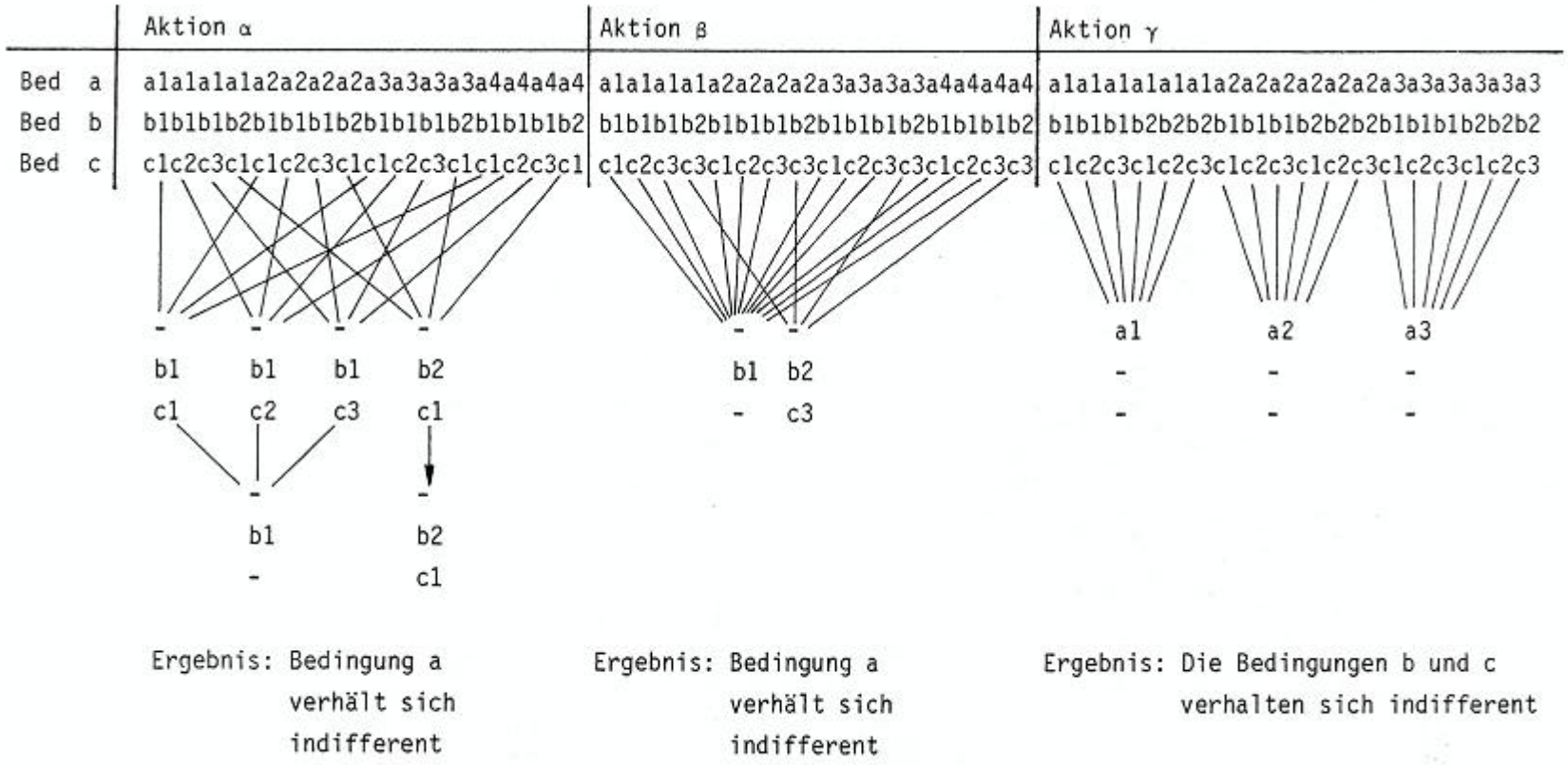
2. Minimieren der Bedingungsanzeigerkombination pro Aktion.

3. Für jede Aktion feststellen, welche Bedingungen nach erfolgter Minimierung als indifferent ausgewiesen sind.

Es empfiehlt sich, zur Relevanzanalyse die Bedingungen und Aktionen matrixförmig anzuordnen.

---

1) Diese vom Verfasser entwickelte Methode wurde zum ersten Mal beschrieben in Strunz (Zergliederung). Vgl. ferner Strunz (Eine Methode zur Zergliederung) und Strunz (Parsing)


Aktion α
Aktion β
Aktion γ
Bed a
Bed b
Bed c
a1a1a1a1a2a2a2a2a3a3a3a3a4a4a4a4
b1b1b1b2b1b1b1b2b1b1b1b2b1b1b1b2
c1c2c3c1c1c2c3c1c1c2c3c1c1c2c3c1
a1a1a1a1a2a2a2a2a3a3a3a3a4a4a4a4
b1b1b1b2b1b1b1b2b1b1b1b2b1b1b1b2
c1c2c3c3c1c2c3c3c1c2c3c3c1c2c3c3
a1a1a1a1a1a1a2a2a2a2a2a2a3a3a3a3a3a3
b1b1b1b2b2b2b1b1b1b2b2b2b1b1b1b2b2b2
c1c2c3c1c2c3c1c2c3c1c2c3c1c2c3c1c2c3
- - - -
b1 b1 b1 b2
c1 c2 c3 c1
- -
b1 b2
- c1
- -
b1 b2
- c3
a1 a2 a3
- - -
- - -
Ergebnis: Bedingung a verhält sich indifferent
Ergebnis: Bedingung a verhält sich indifferent
Ergebnis: Die Bedingungen b und c verhalten sich indifferent

Nach der Relevanzanalyse für die Aktionen steht fest, dass eine Teiltabelle mit den Bedingungen b und c und den Aktionen α und β gebildet werden kann sowie eine weitere Teiltabelle mit der Bedingung a und der Aktion γ Beide Tabellen können sequentiell verknüpft werden, damit, wie in der Ursprungstabelle, die Aktion γ auf die Aktionen α und β folgt.

TAB 1.1			1	2	3	4	5	6
B1	Bedingung b		b1	b1	b1	b2	b2	b2
B2	Bedingung c		c1	c2	c3	c1	c2	c3
A1	Aktion α		α3	α1	α2	α1		
A2	Aktion β		β3	β2	β2			β1
A3	Gehe nach TAB 1.2		X	X	X	X	X	X

TAB 1.2			1	2	3	4
B1	Bedingung a		a1	a2	a3	a4
A1	Aktion γ		γ1	γ2	γ3	

Eine vergleichende Bewertung der vorgestellten Verfahren kann sich an den Merkmalen des jeweils entstehenden Entscheidungstabellenverbunds orientieren. Dabei ist von Interesse, welchen Umfang die syntaktischen Einheiten der ursprünglichen Entscheidungstabelle annehmen und welche Wegestrecken im Verbund bewältigt werden müssen, um die zutreffende Aktionsfolge auszuführen.

Die *Zergliederung durch Regelaufteilung* ist bei jeder Tabelle anwendbar und reduziert die Anzahl der Regeln pro Teiltabelle auf jedes gewünschte Maß. Die Anzahl der Regeln insgesamt bleibt gleich.

Die Betrachtung der syntaktischen Einheiten des entstehenden Entscheidungstabellenverbunds offenbart die Nachteile dieses Zergliederungsverfahrens jedoch recht deutlich: Jede entstehende Teiltabelle enthält jeweils die vollständige Basisfolge von Bedingungen und Aktionen. Sofern sich die gesuchte Regel nicht in der ersten Teiltabelle des Verbundes befindet, beinhaltet die Wegestrecke zur Ermittlung der zutreffenden Aktionsfolge eine mehrfache Überprüfung der Basisfolge von Bedingungen.

Die *Zergliederung durch Bedingungsaufteilung* ist ebenfalls bei jeder Tabelle anwendbar. Die entstehenden Teiltabellen des Verbunds weisen insgesamt mehr syntaktische Einheiten auf als die ursprüngliche Tabelle. Eine Betrachtung der Wegestrecken zur Ermittlung jeder Aktionsfolge zeigt jedoch, dass in keinem Fall eine Mehrfachprüfung von Bedingungen erforderlich ist, so dass im Vergleich zur ursprünglichen Entscheidungsstabelle nur die erforderlichen Verknüpfungsaktionen eine geringfügige Verlängerung der Wegestrecken bewirken.

Die *kombinierte Bedingungs-/Aktionsaufteilung* zeigt das eindrucksvollste Ergebnis. Der entstehende Entscheidungstabellenverbund weist erheblich weniger syntaktische Einheiten auf als die ursprüngliche Entscheidungstabelle, auch die Wegestrecken verlängern sich nicht. Dieses Ergebnis darf jedoch nicht verallgemeinert werden. Wenn die Relevanzanalyse ergeben sollte, dass jede Aktion von allen vorhandenen Bedingungen beeinflusst wird, dann bringt dieses Zergliederungsverfahren keinerlei Vorteile.

Je mehr Bedingungen eine Entscheidungstabelle jedoch hat - und nur bei Entscheidungstabellen mit vielen Bedingungen entsteht ein Zergliederungsbedürfnis -, desto wahrscheinlicher ist es aber, dass nicht alle Bedingungen gemeinsam jede Aktion beeinflussen. Es erscheint deshalb sinnvoll, in jedem Fall zunächst die zuletzt beschriebene Methode anzuwenden und erst bei Versagen dieser Methode auf die Zergliederung durch Bedingungsaufteilung überzugehen.

1.1.2.2.2 Kombination von Entscheidungstabellen

Das Problem der Kombination von Entscheidungstabellen ist in der Literatur bisher nicht behandelt worden. Aus folgenden Gründen wird eine Auseinandersetzung mit diesem Teilproblem dennoch für sinnvoll erachtet: Zum einen werden alle Überlegungen zur Optimierung eines Entscheidungstabellenverbunds an der Frage der Kombination von Einzeltabellen dieses Verbunds nicht vorbeigehen können. Zum anderen kann eine systematische Untersuchung syntaktischer Kombinationsmöglichkeiten von Entscheidungstabellen gerade erst begonnene Forschungsarbeiten zu einem hochinteressanten Thema mit erheblicher praktischer Bedeutung befruchten: der automatischen Erstellung eines Entscheidungstabellenverbunds aus einem bereits bestehenden Programm.[1)]

---

1) Vgl. dazu Cavouras (Conversion), Reinwald (Re-conversion problem) und Reinwald (Progress on automatic conversion)

Für die Kombination von unterschiedlichen Entscheidungstabellen eines Entscheidungstabellenverbunds zu einer einzigen Entscheidungstabelle gibt es 9 mögliche Ausgangssituationen:

	1	2	3	4	5	6	7	8	9
Die Bedingungen der zu kombinierenden Tabellen sind	iden-tisch	identisch	iden-tisch	teil-weise iden-tisch	teil-weise iden-tisch	Teilweise identisch	unter-schied-lich	unter-schiedlich	unter-schied-lich
Die Aktionen der zu kombinierenden Tabellen sind	iden-tisch	teil-weise iden-tisch	unter-schied-lich	identisch	identisch	unter-schied-lich	iden-tisch	Teilweise identisch	unter-schied-lich

Zur Vereinfachung der Erläuterung wird davon ausgegangen, dass nur zwei Entscheidungstabellen zu kombinieren sind, die im Entscheidungstabellenverbund sequentiell hintereinander angeordnet sind. Außerdem sollen keine Verfahren zur syntaktischen Kombination für Fälle aufgezeigt werden, die offensichtlich - betrachtet aus semantischer Sicht - irreal sind.

Fall 1: Identische Bedingungen, identische Aktionen

Die Kombinationsmöglichkeiten richten sich nach der logischen Beziehung der Regeln zueinander.

Wird die Menge der Bedingungsanzeigerkombinationen der ersten Tabelle mit K^S und der zweiten Tabelle mit K^t bezeichnet, dann sind folgende Fallunterscheidungen möglich:

(1) $K^s = K^t$
(2) $K^s \cap K^t \neq \phi$
(3) $K^s \subset K^t$ oder $K^s \supset K^t$
(4) $K^s \cap K^t = \phi$

Der Unterfall (1) ist aus semantischer Sicht unwahrscheinlich. Nur für den theoretischen Grenzfall, dass die den Bedingungsanzeigerkombinationen zugeordneten Aktionsanzeigerkombinationen, bezogen auf jede elementare Bedingungsanzeigerkombination, disjunkt sind, wäre eine logische Ausgangssituation für eine Kombination gegeben.

Die Aktionsanzeiger der zweiten Tabelle würden dann in die erste Tabelle einbezogen, und die zweite Tabelle könnte eliminiert werden. Im Unterfall (2) muss die Schnittmenge der Bedingungsanzeigerkombinationen untersucht werden. Ergänzen sich die zugeordneten Aktionsanzeigerkombinationen, wie bei Unterfall (1) beschrieben, dann können die Aktionsanzeiger der Schnittmenge aus der zweiten Tabelle in die erste übertragen werden. Zusätzlich wird der disjunkte Teil der Regeln aus der zweiten Tabelle der ersten angefügt und die zweite Tabelle eliminiert.

Im Unterfall (3) müssen wieder die Aktionsanzeigerkombinationen der Regeln, die die Teilmenge bilden, mit den Aktionsanzeigerkombinationen der korrespondierenden Regeln der anderen Tabelle verglichen werden.

Bei disjunkten Aktionsanzeigerkombinationen ist eine Zusammenfassung möglich.

Unterfall (4) ist der einzig unproblematische Fall. Alle Regeln der zweiten Tabelle können der ersten angefügt werden, und die zweite Tabelle kann eliminiert werden.

Fall 2: Identische Bedingungen, teilweise identische Aktionen

In Bezug auf die identischen Aktionen wird hier ebenso verfahren wie bei Fall 1. Die Ergänzung der unterschiedlichen Aktionen aus der zweiten Tabelle ist unproblematisch.

Fall 3: Identische Bedingungen, unterschiedliche Aktionen

Zur Kombination geeigneter Fall. Folgendes Kombinationsverfahren lässt sich anwenden:

1. Umwandeln beider Tabellen in ihre kanonische Normalform

2. Kombination von Regeln beider Tabellen mit identischen Bedingungsanzeigerkombinationen. Die Verknüpfungsaktion entfällt.

3. Weitmöglichste Minimierung von Regeln mit identischer Aktionsfolge.

Beispiel:
Entscheidungstabellenverbund

TAB 1		1	2	3	4
B1	Bedingung a	J	J	J	N
B2	Bedingung b	J	J	N	--
B3	Bedingung c	J	N	--	--
A1	Aktion α	X	X		X
A2	Aktion β	X		X	X
A3	Aktion γ	X	X	X	X
A4	Gehe nach TAB 2	X	X	X	X

TAB 2		1	2	3	4
B1	Bedingung a	J	N	N	N
B2	Bedingung b	--	J	N	N
B3	Bedingung c	--	--	J	N
A1	Aktion δ		X	X	X
A2	Aktion ε	X		X	
A3	Aktion ξ	X		X	X

Schritt 1: Umwandlung in die kanonischen Normalformen

TAB 1		1	2	3	4	5	6	7	8
B1	Bedingung a	J	J	J	J	N	N	N	N
B2	Bedingung b	J	J	N	N	J	J	N	N
B3	Bedingung c	J	N	J	N	J	N	J	N
A1	Aktion α	X	X			X	X	X	X
A2	Aktion β	X		X	X	X	X	X	X
A3	Aktion γ	X	X	X	X	X	X	X	X
A4	Gehe nach TAB 2	X	X	X	X	X	X	X	X

TAB 2		1	2	3	4	5	6	7	8
B1	Bedingung a	J	J	J	J	N	N	N	N
B2	Bedingung b	J	J	N	N	J	J	N	N
B3	Bedingung c	J	N	J	N	J	N	J	N
A1	Aktion δ					X	X	X	X
A2	Aktion ε	X	X	X	X			X	
A3	Aktion ξ	X	X	X	X			X	X

Schritt 2: Regelkombination

			1	2	3	4	5	6	7	8
B1	Bedingung a		J	J	J	J	N	N	N	N
B2	Bedingung b		J	J	N	N	J	J	N	N
B3	Bedingung c		J	N	J	N	J	N	J	N
A1	Aktion α		X	X			X	X	X	X
A2	Aktion β		X		X	X	X	X	X	X
A3	Aktion γ		X	X	X	X	X	X	X	X
A4	Aktion δ						X	X	X	X
A5	Aktion ε		X	X	X	X			X	
A6	Aktion ξ		X	X	X	X			X	X

Schritt 3: Regelminimierung

			1	2	3	4	5	6
B1	Bedingung a		J	J	J	N	N	N
B2	Bedingung b		J	J	N	J	N	N
B3	Bedingung c		J	N	--	--	J	N
A1	Aktion α		X	X		X	X	X
A2	Aktion β		X		X	X	X	X
A3	Aktion γ		X	X	X	X	X	X
A4	Aktion δ					X	X	X
A5	Aktion ε		X	X	X		X	
A6	Aktion ξ		X	X	X		X	X

Fall 4: Teilidentische Bedingungen, identische Aktionen

Die Kombination beider Tabellen ist dann möglich, wenn sich aus der beschriebenen Logik keine beabsichtigte mehrfache Ausführung von Aktionen ergibt. Diese Voraussetzung ist jedoch nur in dem theoretischen Grenzfall gegeben, dass die Regeln beider Tabellen so aufgebaut sind, dass bei identischen Anzeigern der identischen Bedingungen die Aktionsanzeigerkombinationen disjunkt sind.

Fall 5: Teilweise identische Bedingungen, teilweise identische Aktionen

Wie bei Fall 4 ist eine Kombination nur dann möglich, wenn sich aus der beschriebenen Logik keine beabsichtigte mehr-

fache Ausführung von Aktionen ergibt. Bei identischen Bedingungsanzeigern der identischen Bedingungen von Regeln beider Tabellen müssen die Aktionsanzeigerkombinationen disjunkt sein. Da diese Voraussetzung bei nur teilweise identischen Aktionen häufiger gegeben sein wird als bei Fall 4, sollen die Teilschritte der Kombination erläutert und an einem Beispiel verdeutlicht werden.

1. Relevanzanalyse für alle Aktionen
   Dabei müssen aus verschiedenen Tabellen stammende Bedingungsanzeigerkombinationen zur gleichen Aktion disjunkt sein. Andernfalls würde die identische Aktion im Verbund zweifach ausgeführt, und eine Kombination ist unzulässig.

2. Zusammenfassen der unterschiedlichen Bedingungen und Aufbau des Bedingungsanzeigerteils in vollständiger kanonischer Normalform.

3. Ergänzen der Aktionen und Aktionsanzeiger unter Berücksichtigung der in der Relevanzanalyse ermittelten logischen Beziehungen zwischen Bedingungen und Aktionen.

4. Regelminimierung

   Beispiel: Entscheidungstabellenverbund

TAB 1		1	2	3	4	5	6	7
B1	Bedingung a	J	J	J	N	N	N	--
B2	Bedingung b	J	J	N	J	J	N	N
B3	Bedingung c	J	N	J	J	N	J	N
A1	Aktion α	X			X		X	X
A2	Aktion β		X					X
A3	Aktion γ		X	X		X	X	
A4	Gehe nach TAB 2	X	X	X	X	X	X	X

TAB 2		1	2	3	4	5	6	7
B1	Bedingung a	J	J	J	N	N	N	N
B2	Bedingung b	J	N	N	J	J	N	N
B3	Bedingung c	--	J	N	J	N	J	N
A1	Aktion δ	X					X	X
A2	Aktion ε	X	X	X	X	X	X	
A3	Aktion ξ		X		X			

1. Schritt: Relevanzanalyse für alle Aktionen und Prüfung, ob die Bedingungsanzeigerkombinationen von TAB 1 und TAB 2 zur identischen Aktion y disjunkt sind.

	Aktion α	Aktion β	Aktion γ von TAB 1	Aktion γ von TAB 2	Aktion δ	Aktion ε
Bed. a	J N N -	J -	J J N N			
Bed. b	J J N N	J N	J N J N	J N N	J J J N N N	J N
Bed. c	J J J N	N N	N J N J	J N N	J N N J J N	N J
Bed. d				- J N	- J N J N J	J J

Die von TAB 1 und TAB 2 stammenden Bedingungsanzeigerkombinationen sind im Verhältnis zueinander logisch disjunkt, so dass eine Kombination der Tabellen möglich ist.

Als Vorbereitung zur Kombination werden die Bedingungsanzeigerkombinationen zu jeder Aktion zunächst minimiert. Dabei wird angenommen, dass sich Bedingung d in Bezug auf die Aktionen der Tabelle 1 und Bedingung a in Bezug auf die Aktionen der Tabelle 2 indifferent verhalten.

	Aktion α	Aktion β	Aktion γ	Aktion δ	Aktion ε
Bed. a	J N N -	J -	J J N N - - -	- - - - - -	- -
Bed. b	J J N N	J N	J N J N J N N	J J J N N N	J N
Bed. c	J J J N	N N	N J N J J N N	J N N J J N	N J
Bed. d	- - - -	- -	- - - - - J N	- J N J N J	J J
	- N -	J -	-	- - -	- -
	J N N	J N	-	- J -	J N
	J J N	N N	-	J N N	N J
	- - -	- -	-	- N J	J J

2. und 3. Schritt: Aufbau der kombinierten Entscheidungstabelle in vollständiger kanonischer Normalform und Ergänzen der Aktionsanzeiger Zeile für Zeile

Kom. TAB		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
B1	Bedingung a	J	J	J	J	J	J	J	J	N	N	N	N	N	N	N	N
B2	Bedingung b	J	J	J	J	N	N	N	N	J	J	J	J	N	N	N	N
B3	Bedingung c	J	J	N	N	J	J	N	N	J	J	N	N	J	J	N	N
B4	Bedingung d	J	N	J	N	J	N	J	N	J	N	J	N	J	N	J	N
A1	Aktion α	X	X					X	X	X	X			X	X	X	X
A2	Aktion β			X	X			X	X							X	X
A3	Aktion γ	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
A4	Aktion δ	X	X	X	X	X	X	X		X	X	X	X	X	X	X	
A5	Aktion ε			X		X						X		X			

4. Schritt: Regelminimierung

Kom. TAB minimiert		1	2	3	4	5	6	7	8	9	10	11
B1	Bedingung a	-	J	J	J	J	N	N	N	N	-	-
B2	Bedingung b	J	J	J	N	N	J	J	N	N	N	N
B3	Bedingung c	J	N	N	J	J	N	N	J	J	N	N
B4	Bedingung d	-	J	N	J	N	J	N	N	N	J	N
A1	Aktion α	X							X	X	X	X
A2	Aktion β		X	X							X	X
A3	Aktion γ	X	X	X	X	X	X	X	X	X	X	X
A4	Aktion δ	X	X	X	X	X	X	X	X	X	X	
A5	Aktion ε		X		X			X		X		

Fall 6: Teilweise identische Bedingungen, unterschiedliche Aktionen

Hier ist eine Kombination sinnvoll und unproblematisch. Es kann das für Fall 5 beschriebene Kombinationsverfahren angewendet werden. Es besteht jedoch keine Gefahr der mehrfachen Ausführung identischer Aktionen, so dass entsprechende Analysen unterbleiben können.

Fall 7: Unterschiedliche Bedingungen, identische Aktionen.

Wenn in einem Entscheidungstabellenverbund zwei hintereinander angeordnete Entscheidungstabellen unterschiedliche Bedingungen, aber identische Aktionen aufweisen, dann ist die damit verbundene zweifache Ausführung von Aktionen vom Problem her erforderlich. Eine Kombination würde die zweifache Ausführung verhindern und ist deshalb unzulässig.

Fall 8: Unterschiedliche Bedingungen, teilweise identische Aktionen.

Auch diese Konstruktion impliziert die zweifache Ausführung von Aktionen. Eine Kombination ist deshalb unzulässig.

Fall 9: Unterschiedliche Bedingungen, unterschiedliche Aktionen.

Da keinerlei Gemeinsamkeiten vorhanden sind, entfallt jedes Motiv für eine Kombination. Die Kombination wäre jedoch syntaktisch möglich, wenn sich jeweils die Bedingungen der einen Tabelle zu den Aktionen der anderen Tabelle indifferent verhalten.

1.1.3 Syntaxprüfungen

Eine Entscheidungstabelle kann einer Syntaxprüfung unterzogen werden, um Verstöße gegen Bildungs- oder Transformationsregeln aufzudecken. Die elementaren syntaktischen Einheiten einer Entscheidungstabelle - Bedingungen und Aktionen - bedürfen dabei keiner Überprüfung, da sie selbst nicht durch Anwendung von Bildungs- oder Transformationsregeln entstanden sind. Vielmehr erstrecken sich die Syntaxprüfungen auf die *Regeln* einer Entscheidungstabelle.

Da innerhalb einer Eintreffer-Entscheidungstabelle nur Entscheidungsregeln erlaubt sind, deren Bedingungsanzeigerkombinationen disjunkt sind, wird ein Prüfungsverfahren benötigt, das Abweichungen von dieser Vor-

schrift diagnostiziert. Es ist in der Entscheidungstabellenlogik unter verschiedenen Bezeichnungen bekannt, wie beispielsweise Redundanzprüfung oder Konsistenzprüfung. Nach einer häufig in der Literatur zu findenden Auffassung sind Entscheidungstabellen, deren Bedingungsanzeigerkombinationen nicht disjunkt sind, mehrdeutig.[1)] Die entsprechende Syntaxprüfung, die solche Mehrdeutigkeiten ermittelt, soll deshalb im Folgenden "Eindeutigkeitsprüfung" genannt werden.

Eine Syntaxprüfung kann jedoch auch mit der Zielsetzung erfolgen, festzustellen, ob alle elementaren Bedingungsanzeigerkombinationen durch die Regeln der Entscheidungstabelle repräsentiert sind. Das entsprechende Prüfungsverfahren ist in der Entscheidungstabellentechnik unter der Bezeichnung "Vollständigkeitsprüfung" bekannt.

Unter rein syntaktischen Gesichtspunkten kann die Syntaxprüfung eines *Entscheidungstabellenverbunds* auf die Syntaxprüfung aller enthaltenen Einzeltabellen zurückgeführt werden. Demnach gilt ein Entscheidungstabellenverbund als syntaktisch vollständig und eindeutig, wenn alle enthaltenen Einzeltabellen vollständig und eindeutig sind.

1.1.3.1 Eindeutigkeitsprüfung

Das einfachste Verfahren, Mehrdeutigkeiten in den Entscheidungsregeln einer Entscheidungstabelle zu erkennen, wäre eine Umwandlung der Tabelle in ihre kanonische Normalform. Entstehen dabei auch identische elementare Bedingungsanzeigerkombinationen, dann ist der Mehrdeutigkeitsnachweis erbracht.

Nur bei relativ kleinen Entscheidungstabellen wäre dieses Verfahren jedoch praktikabel. Bereits wenn die Anzahl der entstehenden elementaren Bedingungsanzeigerkombinationen 30 - 40 übersteigt, ist das Erkennen identischer Kombinationen schwierig.

Wesentlich praktikabler ist das im Folgenden beschriebene Verfahren, das in drei Schritten abgewickelt wird:

1. Diagnose der Mehrdeutigkeit durch paarweises Vergleichen der Regeln im Bedingungsanzeigerteil. Wenn nicht in mindestens einer Bedingungszeile disjunkte Bedingungsanzeiger erkannt werden, sind auch die Anzeigerkombinationen nicht disjunkt.[2)]

---

1) Vgl. beispielsweise King (Ambiguity), S. 680

2) Für begrenzte Entscheidungstabellen wurde diese Art der Diagnostik beschrieben von Pollack (Analysis), S. 20 f.

2. Für nicht disjunkte Regelpaare wird durch Bildung des Durchschnitts die gemeinsam enthaltene Bedingungsanzeigerkombination ermittelt.[1)]

3. Die gemeinsam enthaltene Bedingungsanzeigerkombination wird aus einer der beiden Regeln eliminiert.

Beispiel zur Eindeutigkeitsprüfung:

Zu den Bedingungen
Bedingung a mit [a]={a1,a2, a3}
Bedingung b mit [b] ={b1, b2}
Bedingung c mit [c] ={J,N}

Seien folgende Regeln formuliert:

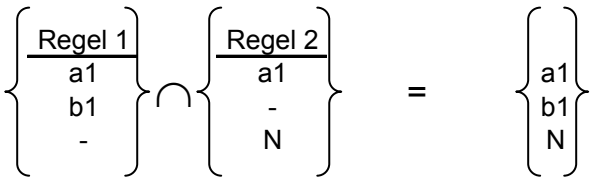
		1	2	3	4	5	6
B1	Bedingung a	a1	a1	a2	a3	a3	-
B2	Bedingung b	b1	-	-	b1	b1	b2
B3	Bedingung c	-	N	-	J	N	-
A1	Aktion α			X		X	X
A2	Aktion β	X		X	X		X
A3	Aktion γ		X				X

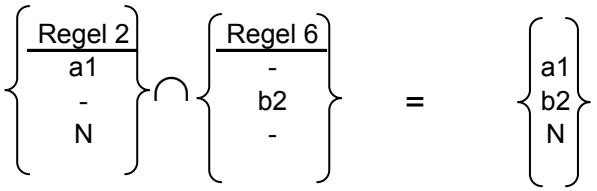
Für jedes Regelpaar wird nun die Beziehung Exklusion verifiziert.
Die Analyse ergibt folgendes Ergebnis

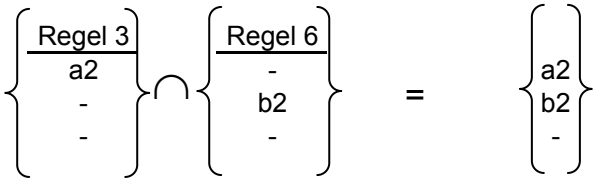
Vergleich der Regeln	Bedingungsanzeiger-kombinationen sind disjunkt	Bedingungsanzeiger-kombinationen sind nicht disjunkt
1 mit 2		X
1 mit 3	X	
1 mit 4	X	
1 mit 5	X	
1 mit 6	X	
2 mit 3	X	
2 mit 4	X	
2 mit 5	X	
2 mit 6		X
3 mit 4	X	
3 mit 5	X	
3 mit 6		X
4 mit 5	X	
4 mit 6	X	
5 mit 6	X	

1) Für begrenzte und erweiterte Entscheidungstabellen beschrieben in Strunz (Handbuch 1972), Abschnitt 5.1. Vgl. dazu ebenso Montalbano (Decision tables), S. 80. Montalbano wendet konsequent das Instrumentarium der Mengenlehre an.

Durch eine Durchschnittbildung kann die Ursache für die fehlende Exklusion der Anzeigerkombinationen näher bestimmt werden. Der Durchschnitt der Bedingungsanzeigerkombinationen ergibt sich aus dem Durchschnitt der paarweise einander gegenüberstehenden Bedingungsanzeiger.







Zur Beseitigung der Mehrdeutigkeit muss die ermittelte gemeinsame Bedingungsanzeigerkombination aus einer der beiden betroffenen Regeln eliminiert werden. Dabei lassen sich zwei Unterfälle der Mehrdeutigkeit unterscheiden. Enthalten die betroffenen Regeln identische Aktionsfolgen, dann liegt *Redundanz* vor. Die Auswahl der zu korrigierenden Regel kann willkürlich erfolgen. Enthalten die betroffenen Regeln unterschiedliche Aktionsfolgen, dann wird diese Art von Mehrdeutigkeit als *Widerspruch* näher gekennzeichnet. In diesem Fall ist keine Korrektur auf rein syntaktischer Basis möglich. Erst wenn über eine semantische Prüfung die fehlerhafte Regel bestimmt ist, kann auf Syntaxebene korrigiert werden.

Die Korrektur einer der beiden Regeln erfolgt in der Weise, dass die als gemeinsam erkannte Bedingungsanzeigerkombination von der Bedingungsanzeigerkombination der zu korrigierenden Regel abgezogen wird. Diese Mengenoperation kann am besten auf der Basis elementarer Bedingungsanzeigerkombinationen erfolgen. Soll beispielsweise die Redundanz zwi-

schen Regel 2 und Regel 6 durch Korrektur der Regel 6 beseitigt werden, dann werden folgende Transformationen ausgeführt:

(1)

Regel 6
-
b2
-

äqsyn

Regel 6					
a1	a1	a2	a2	a3	a3
b2	b2	b2	b2	b2	b2
J	N	J	N	J	N

(2)

Regel 6					
a1	a1	a2	a2	a3	a3
b2	b2	b2	b2	b2	b2
J	N	J	N	J	N

−

a1
b2
N

=

Korr. Regel 6				
a1	a2	a2	a3	a3
b2	b2	b2	b2	b2
J	J	N	J	N

äqsyn

Korr. Regel 6		
a1	a2	a3
b2	b2	b2
J	-	-

1.1.3.2 Vollständigkeitsprüfung

Zwei Verfahren zur Vollständigkeitsprüfung von Entscheidungstabellen sind in der Entscheidungstabellentechnik möglich: der Vergleich der Entscheidungstabelle mit ihrer vollständigen konjunktiven Normalform und die rechnerische Vollständigkeitsprüfung.

Das erste Verfahren verlangt zunächst die Erstellung einer syntaktisch äquivalenten Entscheidungstabelle in vollständiger kanonischer Normalform auf der Basis der Bedingungen der zu überprüfenden Tabelle. Unvollständigkeiten der zu überprüfenden Tabelle können dann dadurch ermittelt werden, dass zu jeder Regel dieser Tabelle die syntaktischen Äquivalenzen in der Normalform-Tabelle kenntlich gemacht werden. Regeln der Normalform-Tabelle, die unbezeichnet bleiben, enthalten die in der zu überprüfenden Tabelle fehlenden elementaren Bedingungsanzeigerkombinationen.

Dieses Verfahren ist nicht sehr praktikabel, wenn die Anzahl der· elementaren Bedingungsanzeigerkombinationen sehr groß ist, weil für die Erstellung der Normalform-Tabelle und die Kennzeichnung der syntaktischen Äquivalenzen zuviel Aufwand entsteht.

Die Praxis der Entscheidungstabellen-Anwendung bevorzugt deshalb ein rechnerisches Verfahren.[1)] Auf der Basis der Bedingungen und ihrer zugehörigen Bedingungsanzeiger wird die Anzahl der elementaren Bedingungsanzeigerkombinationen errechnet. Danach wird errechnet, wie viele elementare Bedingungsanzeigerkombinationen durch die Regeln der zu prüfenden Tabelle repräsentiert sind. Durch Vergleich beider Ergebnisse kann die Vollständigkeit einer Entscheidungstabelle schlüssig nachgewiesen werden. Generelle Voraussetzung für dieses Verfahren ist jedoch, dass die Bedingungsanzeigerkombinationen der Regeln der zu prüfenden Tabelle disjunkt sind. Die Eindeutigkeitsprüfung muss also bereits stattgefunden haben.

Wird innerhalb einer Entscheidungstabelle der Ausschlußanzeiger verwendet, dann kann das beschriebene Verfahren beibehalten werden, wenn der Ausschlußanzeiger formal wie ein Indifferenzanzeiger behandelt wird.

Sollen auch fehlende Bedingungsanzeigerkombinationen ermittelt werden, dann wird das folgende Verfahren anwendbar, das für die zu überprüfende Tabelle Zeile für Zeile die fehlenden Bedingungsanzeigerkombinationen ermittelt. Das Verfahren lässt sich am besten anhand eines Beispiels anschaulich erklären.

Zur folgenden Entscheidungstabelle sollen die nicht in den Regeln enthaltenen Bedingungsanzeigerkombinationen ermittelt werden:

		1	2	3	4	5	6
B1	Bedingung a	J	J	J	J	N	N
B2	Bedingung b	-	-	-	-	J	-
B3	Bedingung c	3	1	3	2	1	-
B4	Bedingung d	J	-	N	-	J	N

1) Vgl. dazu Pollack (Analysis), S. 10 ff.

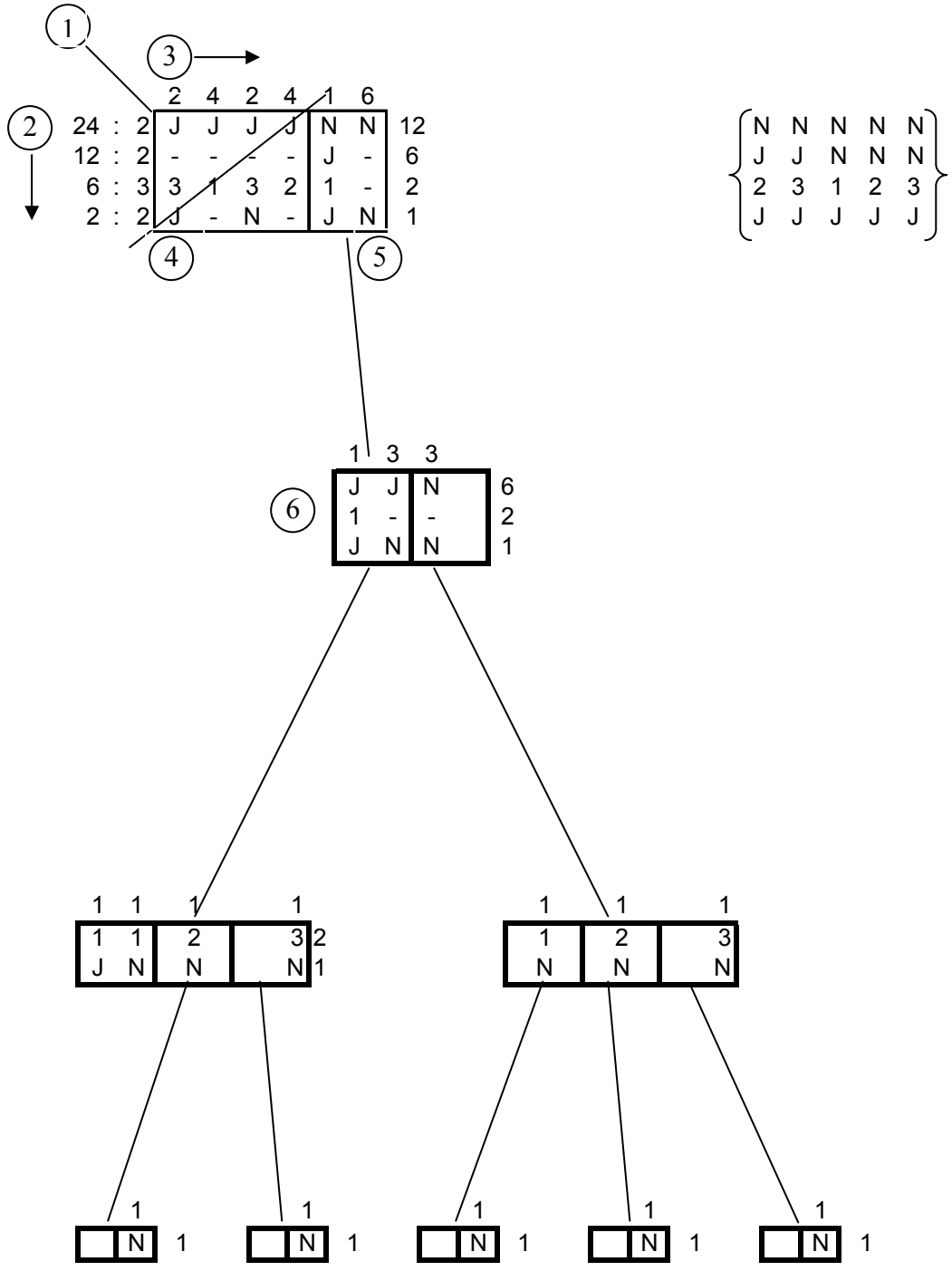


Abb. 3: Ermittlung fehlender Bedingungsanzeiger -Kombinationen

(1) Der Bedingungsanzeigerteil der zu analysierenden Tabelle wird in Abhängigkeit von den Bedingungsanzeigern in der obersten Bedingung in mehrere Teile unterteilt, die getrennt auf Vollständigkeit geprüft werden. Da die oberste Bedingung der Beispieltabelle über die beiden Anzeiger J und N verfügt, entstehen zwei Bereiche.

(2) In jeder Zeile wird zunächst die Anzahl der Bedingunganzeiger vermerkt.
Durch Bilden des Produkts wird die Anzahl der maximal möglichen Bedingungsanzeigerkombinationen ermittelt (24), 24 wird durch die Anzahl der Bedingungsanzeiger der obersten Bedingung geteilt (24:2) und es ergibt sich 12. Dieser Quotient wird rechts neben die Tabelle in Höhe der obersten Bedingung eingetragen. Er gibt an, wie viele Bedingungsanzeigerkombinationen in einer vollständigen Tabelle J und wie viele N als Anzeiger der obersten Bedingung haben würden. Die Zahl 12 wird ebenfalls als Dividend in der zweiten Zeile benutzt, der entsprechende Quotient 6 wird wieder rechts eingetragen usw ..

(3) Über jede Regel wird eingetragen, wie viele einfache Bedingungsanzeigerkombinationen sie enthält.

(4) Durch Addition der in den Regeln der linken Tabellenhälfte enthaltenen Bedingungsanzeigerkombinationen (2+4+2+4) ergibt sich 12. Die linke Tabellenhälfte ist also vollständig. Sie wird durchgestrichen, um anzuzeigen, dass sie nicht mehr weiter analysiert werden muss.

(5) Durch Addition der in der Regel der rechten Tabellenhälfte enthaltenen Bedingungsanzeigerkombinationen (1+6) ergibt sich 7. Damit fehlen 5 Bedingungsanzeigerkombinationen mit N als Anzeiger der obersten Bedingung. 5 N-Anzeiger werden rechts neben der Tabelle der obersten Zeile der fehlenden Anzeigerkombinationen eingetragen.

(6) Die oberste Bedingungszeile der unvollständigen rechten Tabellenhälfte kann nunmehr entfallen und die Analyse der zweiten Bedingungszeile nach dem gleichen Verfahren beginnen. Da auch die zweite Bedingung der Tabelle 2 Anzeiger aufweist, werden wiederum 2 Bereiche gebildet und die bereits vorhandenen Bedingungsanzeigerkombinationen auf die Bereiche aufgeteilt. Dabei ist es erforderlich, den Irrelevanzanzeiger der letzten Regel aufzulösen. Die Vollständigkeitsberechnung ergibt, dass 2 Anzeigerkombinationen mit J und 3 Anzeigerkombinationen mit N fehlen. "J J N N N" wird deshalb als zweite Zeile der fehlenden Anzeigerkombinationen eingetragen.

(7) Das gleiche Verfahren wird auch auf die dritte und vierte Zeile der Entscheidungstabelle angewendet und es ergeben sich " 2 3 1 2 3 " und " J J J J J " als weitere Zeilen der fehlenden Anzeigerkombinationen. Die Ermittlung der fehlenden Anzeigerkombinationen ist damit beendet.

1.2 Semantik der Entscheidungstabelle

Während unter ausschließlich *syntaktischen Gesichtspunkten* die formale Beziehung der elementaren syntaktischen Einheiten "Bedingungen" und "Aktionen" im Rahmen der Gestaltung der höheren syntaktischen Einheiten "Anzeigerkombinationen", "Regeln", "Tabellen", "Tabellenverbund" im Vordergrund stand, untersucht die Semantik die Beziehungen der Entscheidungsstabellen bzw. Entscheidungstabellenelemente zum Objekt, das sie beschreiben, dem Entscheidungsfeld.

Ein *Entscheidungsfeld* repräsentiert alle Daten, die das Handeln eines Entscheidungssubjektes beeinflussen, es bildet somit seine relevante Umwelt.[1)]. Als *Entscheidungssubjekt* soll hier jede Einheit verstanden werden, die zur Aufnahme, Transformation und Abgabe von Daten in der Lage ist. Sowohl der Mensch als auch informationsverarbeitende Maschinen kommen nach dieser Definition als Entscheidungssubjekt infrage.[2)]

Entscheidungen, die von maschinellen Entscheidungssubjekten ausgeführt werden sollen, bedürfen jedoch einer besonders detaillierten Konzeption und Beschreibung. Zur Automatisierung eignen sich nur Entscheidungen, die unter der Prämisse vollkommener Information ausgeführt werden. Sie werden auch als programmierbare Entscheidungen bezeichnet.[3)]

---

1) Auf eine ausführliche Ableitung des Begriffes "Entscheidungsfeld" kann hier verzichtet werden.
Vgl. dazu Frese (Kontrolle), S. 27 ff.

2) Diese Definition befindet sich nicht in Übereinstimmung mit einem umfangreichen Teil der Literatur zur Entscheidungstheorie. Vielfach wird die Auffassung vertreten, dass nur rational handelnde Individuen, die über ein Wertsystem verfügen, Entscheidungen treffen können. Daraus wird abgeleitet, dass nur Menschen die Funktionen von Entscheidungssubjekten übernehmen können.

3) Simon (Management decision), S. 18 ff

1.2.1 Abbildungsfunktion der Entscheidungstabelle

Entscheidungstabellen sind Hilfsmittel zur Abbildung von Entscheidungsfeldern. Abgebildet werden statische und dynamische Merkmale des Entscheidungsfeldes. Die wichtigsten Anforderungen an die Abbildungsfunktion der Entscheidungstabelle sind Eindeutigkeit und Umkehrbarkeit.

1.2.1.1 Abbildung der Aufbaustruktur des Entscheidungsfeldes

Als wichtigste Bestandteile der Aufbaustruktur eines Entscheidungsfeldes können Bedingungen, Aktionen und Entscheidungsregeln angesehen werden.

Die *Bedingungen* beeinflussen die Auswahl der Handlungsalternativen und müssen vom Entscheidungssubjekt auf ihren Zustand überprüft werden.

Die *Aktionen* beschreiben die Handlungsalternativen des Entscheidungssubjektes.

Die *Entscheidungsregeln* sind das statische Abbild antizipierter Transformations- und Bewertungsprozesse. Jede Entscheidungsregel ordnet einer Bedingungskonstellation eine Aktionsfolge zu und macht so eine Analyse möglicher Transformationen des Entscheidungsfeldes und ihre Bewertung überflüssig.

In drei korrespondierenden Bereichen werden die genannten Bestandteile der Aufbaustruktur des Entscheidungsfeldes abgebildet.

Der *Bereich der Bedingungen* enthält die Basisfolge der Bedingungen und bildet damit alle entscheidungsrelevanten Bedingungen ab.

Der *Bereich der Aktionen* enthält die Basisfolge der Aktionen und bildet damit alle zur Beschreibung der Handlungsalternativen benötigten Aktionen ab.

Der *Bereich der Entscheidungsregeln* bildet Bedingungskonstellationen und Aktionsfolgen ab und ordnet sie einander zu.

1.2.1.1.1 Abbildung der Bedingungen

Die zur Beschreibung von Bedingungen und Bedingungsanzeigern benutzten syntaktischen Formen genügen auch den semantischen Anforderungen, wenn die Bedingungsanzeiger so gebildet werden, dass sie für den Entscheidungsprozeß relevante Bedingungszustände abbilden. In aussagenlogischer Betrachtung kann die Bedingung als Aussage aufgefasst werden, wenn ihr aufgrund des Bedingungsanzeigers ein definierter Zustand zugewiesen wurde.

Eine *elementare Bedingungsanzeigerkombination* wird im Rahmen der Abbildung dann benötigt, wenn eine zuzuordnende Aktionsfolge verlangt, dass als Voraussetzung für ihre Ausführung bei jeder Bedingung der Basisfolge genau einer der insgesamt zulässigen Bedingungszustände vorliegt.

Auch für den Fall, dass nicht alle Bedingungen der Basisfolge immer gemeinsam die Handlungsalternativen bestimmen, bestehen syntaktische Hilfsmittel: Indifferenzanzeiger und Ausschlußanzeiger.

In seiner semantischen Funktion beschreibt der *Indifferenzanzeiger* "_" eine Bedingung der Basisfolge als indifferent in Bezug auf die Ausführung der zugeordneten Aktionsfolge. Die semantische Bedeutung des Indifferenzanzeigers ist demnach umfassender als seine syntaktische Funktion. Während er in seiner syntaktischen Funktion alle anderen explizit formulierten Bedingungsanzeiger repräsentiert, beschreibt er in seiner semantischen Funktion alle *Bedingungszustände* der betreffenden Bedingung, also auch Bedingungszustände, die möglicherweise nicht durch die definierten Bedingungsanzeiger erfasst sind. Die korrekte Anwendung des Indifferenzanzeigers verlangt also eine semantische Verifikation der explizit formulierten Bedingungsanzeiger.

Der *Ausschlußanzeiger* "#" wird unter semantischen Aspekten dann sinnvoll angewendet, wenn aufgrund logischer Abhängigkeiten der Bedingungen untereinander bei einer aus einer Teilmenge der Bedingungen der Basisfolge gebildeten Bedingungskonstellation die Überprüfung der anderen Bedingungen der Basisfolge undefinierte Ergebnisse bringt, die die Auswahl einer Entscheidungsregel unmöglich machen.

1.2.1.1.2 Abbildung der Aktionen

Auch zur Beschreibung der Aktionen, die insgesamt die Handlungsalternativen im Entscheidungsfeld definieren, genügen die syntaktischen Möglichkeiten den semantischen Anforderungen.

Da nicht immer alle Aktionen der Basisfolge zur Definition einer Aktionsfolge benötigt werden, besteht die semantische Notwendigkeit des Ausschlusses von Aktionen. Der im Rahmen der Syntax der Entscheidungsstabelle definierte *Ausschlußanzeiger* erfüllt auch in semantischer Sicht diese Funktion.

1.2.1.1.3 Abbildung der Entscheidungsregeln

In semantischer Betrachtung besteht eine Entscheidungsregel aus der Zuordnung einer Aktionsfolge zu einer Bedingungskonstellation. Zur Abbildung einer *Bedingungskonstellation* ist die Bedingungsanzeigerkombination die geeignete syntaktische Einheit. Die Verknüpfung der Bedingungen einer Bedingungskonstellation untereinander ist das logische UND. Nur seltene Ausnahmen in der Literatur stellen auch die Inklusiv-ODER-Verknüpfung der Bedingungen als Alternative heraus.[1)]

Zur Abbildung einer *Aktionsfolge* dient die syntaktische Einheit "Aktionsanzeigerkombination". Die Verknüpfung mehrerer Aktionen zu einer Aktionsfolge wird durch das logische UND realisiert.

Die logische Verknüpfung mehrerer Entscheidungsregeln einer Entscheidungstabelle ergibt sich als Resultat der logischen Beziehungen der abgebildeten Bedingungskonstellationen. Wenn alle abgebildeten Bedingungskonstellationen logisch disjunkt sind, ergibt sich als Beziehung der Regeln zueinander das exklusive ODER. Jede Abweichung davon gilt innerhalb der Eintreffer-Entscheidungstabelle als Mehrdeutigkeit.[2)]

1.2.1.2 Abbildung der Ablaufstruktur des Entscheidungsfeldes

Charakteristisch für programmierte Entscheidungen ist, dass ihr Ablauf nach festen Regeln erfolgt. Ein Entscheidungssubjekt, das eine programmierte Entscheidung ausführt, findet in seinem Entscheidungsfeld immer eine einzige Handlungsalternative für die gegebene Ausgangssituation vor, so dass sich der Entscheidungsprozeß auf die folgenden Phasen reduziert:

---

1) Vgl. beispielsweise Baun (Entscheidungstabellen), S. 5
Vorteilhaft an der Verwendung der Inklusiv-ODER-Beziehung ist, dass in manchen Fällen weniger Regeln entstehen als bei UND-Beziehung. Der Aufbau des Bedingungsanzeigerteils ist jedoch nicht mehr so einfach syntaktisch überprüfbar und verifizierbar wie bei UND-Verknüpfung der Bedingungen.
Die logischen Auswirkungen der Inklusiv-ODER-Beziehung der Bedingungen und die gemeinsame Verwendung von Regeln mit UND-Verknüpfung und Regeln mit ODER-Verknüpfung der Bedingungen innerhalb einer Entscheidungstabelle werden bei Pollack beschrieben. Vgl. Pollack (Analysis), S. 25 ff.
Da die ODER-Beziehung keinerlei pragmatische Bedeutung in der derzeitigen Praxis der Anwendung der Entscheidungstabellentechnik hat, kann auf eine ausführliche Darstellung verzichtet werden.

2) Im Gegensatz dazu lässt die Mehrtreffer-Entscheidungstabelle das Inklusiv-ODER der Regeln zu.
Vgl. dazu Abschnitt 2

Erfassung der Ausgangssituation

Unter der Voraussetzung, dass das Entscheidungsziel bekannt ist, besteht die Erfassung der Ausgangssituation darin, die Zustände der einflußnehmenden Bedingungen festzustellen.

Auswahl der zutreffenden Entscheidungsregel

Ist die aktuelle Bedingungskonstellation bekannt, ist damit jeder Bedingung ein Wahrheitswert zugewiesen. Jede Regel entspricht einer Verknüpfung von Bedingungen, so dass auch jeder Regel ein Wahrheitswert zugewiesen werden kann. Ausgewählt wird diejenige Regel, die den Wahrheitswert WAHR besitzt.

Ausführung der Aktionen

Durch die innerhalb der zutreffenden Entscheidungsregel beschriebene Aktionsfolge ist die auszuführende Handlungsalternative klar definiert. Mit der Ausführung der angezeigten Aktionen findet der Entscheidungsprozeß seinen Abschluss.

Die Ausführung des dargestellten Ablaufs wird durch folgende Eigenschaften der Entscheidungstabelle erleichtert oder erst ermöglicht:

Das Entscheidungsziel kann aus der Bezeichnung der Entscheidungstabelle hervorgehen.

Die einflußnehmenden Bedingungen sind im linken oberen Quadranten der Entscheidungstabelle aufgeführt. Sie sind in der aufgeführten Reihenfolge von oben nach unten prüfbar.

Die möglichen Entscheidungsregeln sind parallel nebeneinander angeordnet und werden damit auch optisch als Alternativen ausgewiesen. Jede Entscheidungsregel bildet mit der enthaltenen Bedingungsanzeigerkombination diejenige Bedingungskonstellation eindeutig ab, für die die zugeordnete Aktionsfolge ausgeführt werden soll.

Die Ausführung der angezeigten Aktionen erfolgt in der innerhalb der Tabelle aufgeführten Reihenfolge - von oben nach unten. Mit der Ausführung der letzten in der Reihenfolge angezeigten Aktion ist der Entscheidungsprozeß beendet.

Umfangreiche Entscheidungsprozesse sind durch eine große Anzahl zu prüfender Bedingungen und auszuführender Aktionen, durch die Möglichkeit einer mehrfachen Prüfung von Bedingungen und Ausführung von Aktionen sowie durch die Notwendigkeit gekennzeichnet, bestimmte Aktionen zeit-

lich vor bestimmten Bedingungen auszuführen, da sie diese Bedingungen beeinflussen. In diesem Fall ist der Entscheidungstabellenverbund die geeignete Beschreibungsform. Ein Entscheidungstabellenverbund kann so konstruiert werden, dass er auch den Anforderungen sehr komplexer Entscheidungsprozesse gerecht wird.

1.2.2 Verifikation der Entscheidungstabelle

Der Verifikation einer Entscheidungstabelle dienen alle Verfahren, die geeignet sind, den objektiven Nachweis zu führen, dass die tabellarische Abbildung des Entscheidungsfeldes den relevanten Merkmalen des realen Entscheidungsfeldes isomorph ist und ein auf der Basis der Entscheidungstabelle vollzogener Entscheidungsprozeß das gewünschte Ergebnis erbringt.

Unter der Prämisse einer vollkommenen Information über das reale Entscheidungsfeld und bei Ausschaltung subjektiver Einflüsse des Entscheidungssubjekts sowie von Mängeln der zur Abbildung des Entscheidungsfeldes benutzten Sprache wäre das Problem fehlerhafter Entscheidungstabellen nicht existent. In der Praxis der Gestaltung rechnergestützter Informationssysteme jedoch bemühen sich Datenverarbeitungsspezialisten und Fachabteilungsangehörige, die keine gemeinsame "Sprache" sprechen, antizipierte Entscheidungsprozesse zu beschreiben, von denen sie nur unvollkommene Informationen besitzen. Aufgrund ihrer subjektiven Unzulänglichkeit sind sie sich jedoch dieser Unvollkommenheit häufig nicht bewusst. Aufgabe der Entscheidungstabellen-Verifikation ist es demnach, durch systematische Prüfungen solche Teile der Entscheidungstabelle zu lokalisieren, die relevante Merkmale des realen Entscheidungsfeldes fehlerhaft abbilden.

Nur zu verständlich ist es, dass die Entscheidungstabellen-Literatur tendenziell versucht, die Verifikation einer Entscheidungstabelle als rein syntaktisches Problem einzustufen. Die syntaktische Dimension der Entscheidungstabelle umfasst klar definierte Bildungs- und Transformationsregeln, deren korrekte Einhaltung sich leicht nachprüfen lässt. Ohne Zweifel haben syntaktische Prüfverfahren auch eine enorme Bedeutung im Rahmen der semantischen Verifikation, sie werden jedoch in ihren Auswirkungen häufig überschätzt und bieten dem Entscheidungstabellenwender in vielen Fällen nur eine trügerische Sicherheit, sein Problem korrekt beschrieben zu haben.

Das wichtigste Anliegen der folgenden Ausführungen zur Verifikation von Entscheidungstabellen ist es, den Leistungsumfang syntaktischer Prüfverfahren klar abzugrenzen und die Notwendigkeit zusätzlicher Verifikationsprozeduren zu begründen.

1.2.2.1 Analyse semantischer Redundanzen und Mehrdeutigkeiten

Redundanzen und Mehrdeutigkeiten entstehen in einer Entscheidungstabelle, wenn Teile des Entscheidungsfeldes mehrfach abgebildet werden, die nicht entscheidungsrelevant sind. Die Analyse dieser Abbildungsfehler erstreckt sich auf die Basisfolgen von Bedingungen und Aktionen sowie auf die Entscheidungsregeln.

1.2.2.1.1 Analyse der Basisfolge der Bedingungen

Eine in der Basisfolge der Bedingungen einer Entscheidungstabelle enthaltene Bedingung ist redundant, wenn ihre Überprüfung keine der auszuführenden Aktionsfolgen beeinflusst. Zur Analyse redundanter Bedingungen kann eine auf die abgebildeten Aktionsfolgen bezogene Relevanzanalyse erforderlich sein.

Beispiel:

		1	2	3	4
B1	Bedingung a	J	J	J	N
B2	Bedingung b	J	J	N	-
B3	Bedingung c	J	N	-	-
A1	Aktion α			X	X
A2	Aktion β	X	X		X
A3	Aktion γ	X	X	X	

Die ersten beiden Regeln enthalten eine identische Aktionsfolge. Relevante Bedingungsprüfungen zur Auslösung dieser Aktionsfolge sind jedoch nur die Prüfungen der Bedingungen a und b. Wenn die Bedingungen a und b beide erfüllt sind, wird ohne Rücksicht auf den Zustand der Bedingung c die Aktionsfolge ausgeführt.

Auch syntaktisch kann dieser semantische Inhalt der Entscheidungstabelle einfach veranschaulicht werden, da sich die ersten beiden Regeln der Tabelle zu einer einzigen Regel zusammenfassen lassen. Nach der syntaktischen Minimierung ist auch äußerlich erkennbar, dass die Bedingung c redundant ist.

		1	2	3
B1	Bedingung a	J	J	N
B2	Bedingung b	J	N	-
B3	Bedingung c	-	-	-
A1	Aktion α		X	X
A2	Aktion β	X		X
A3	Aktion γ	X	X	

1.2.2.1.2 Analyse der Basisfolge der Aktionen

Eine in der Basisfolge der Aktionen einer Entscheidungstabelle enthaltene Aktion ist redundant, wenn sie nicht Bestandteil mindestens einer der in dieser Entscheidungstabelle beschriebenen Aktionsfolgen ist. Eine redundante Aktion kann bereits syntaktisch daran erkennbar sein, dass Aktionsanzeiger völlig fehlen.

Schwieriger zu analysieren sind Aktionsredundanzen, die syntaktisch nicht ohne weiteres erkennbar sind und sich erst in der dynamischen Ausführung des Entscheidungsprozesses auf der Basis der Entscheidungsstabelle auswirken. Solche Aktionsredundanzen entstehen beispielsweise, wenn eine Aktion in ihrer Ausführung an eine logisch unzulässige Bedingungskonstellation gebunden ist.

		1	2	3
B1	Alter <= 21	J	N	N
B2	Alter > 21	N	J	N
A1	Aktion α			X
A2	Aktion β		X	X
A3	Aktion γ	X		X

ist durch die beiden Bedingungen Bl und B2 der gesamte Wertebereich für "Alter" abgebildet. Dass weder Bl noch B2 in einem konkreten Fall zutreffen, ist logisch nicht vorstellbar. Die Bedingungsanzeigerkombination der Regel 3 beschreibt also eine Bedingungskonstellation, die niemals Bestandteil des Entscheidungsfeldes sein wird. Da aber Aktion α nur innerhalb der Regel 3 relevant ist, kann sie ebenso wie die gesamte Regel 3 als redundant bezeichnet werden.

Aktionsredundanzen der dargestellten Art sind nur als Nebenergebnis einer Analyse erkennbar, die die vollständige Abbildung aller logisch zulässigen Bedingungskonstellationen durch die Entscheidungstabelle überprüft.

1.2.2.1.3 Analyse der Entscheidungsregeln

Das bereits beschriebene syntaktische Verfahren zur Eindeutigkeitsprüfung [1)] erfüllt im Rahmen semantischer Analysen nicht alle Anforderungen. Ein Beispiel soll das verdeutlichen:

Die beiden Bedingungsanzeigerkombinationen

$$\left\{\begin{array}{l} > 5 \\ = 7 \\ \quad N \\ \quad A \end{array}\right\} \qquad \left\{\begin{array}{l} > 5 \\ = 7 \\ \quad N \\ \quad B \end{array}\right\}$$

enthalten - syntaktisch analysiert - keine Mehrdeutigkeit. Syntaktisch unterschiedliche Bedingungsanzeiger in der letzten Bedingungszeile lassen die Bedingungsanzeigerkombinationen disjunkt erscheinen. Die semantische Eindeutigkeitsanalyse hat jedoch die konkrete Bedeutung der Bedingungsanzeiger in die Betrachtung einzubeziehen. Angenommen, "A" und "B" sind symbolische Namen für Datenelemente, die auch identische Werte annehmen können. Dann ergibt sich aber eine semantische Mehrdeutigkeit, die sogar einen logischen Widerspruch beinhalten kann, wenn den beiden Bedingungsanzeigerkombinationen unterschiedliche Aktionsfolgen zugeordnet sind.

Zur Korrektur dieser semantischen Mehrdeutigkeit muss der gemeinsame Abbildungsbereich der Variablen ermittelt und aus einer der beiden·Variab-

---

1) Vgl. dazu Abschnitt 1.1.3.1

len eliminiert werden. Sollte sich das nicht durchführen lassen, ist eine Umwandlung der letzten Bedingung in zwei einfache Bedingungen erforderlich, um die semantische Mehrdeutigkeit ausschalten zu können. Es entstehen dann drei Bedingungsanzeigerkombinationen:

	>5	>5	>5
	=7	=7	=7
	N	N	N
A:	J	N	J
B:	N	J	J

In der Regel wird semantische Mehrdeutigkeit jedoch mit einer leicht modifizierten Form der syntaktischen Eindeutigkeitsprüfung diagnostiziert und entfernt werden können.

Aufgrund ihrer erheblichen praktischen Bedeutung soll die semantische Eindeutigkeitsprüfung an einem Beispiel demonstriert werden. Folgende Entscheidungstabelle soll auf semantische Mehrdeutigkeiten hin geprüft werden:

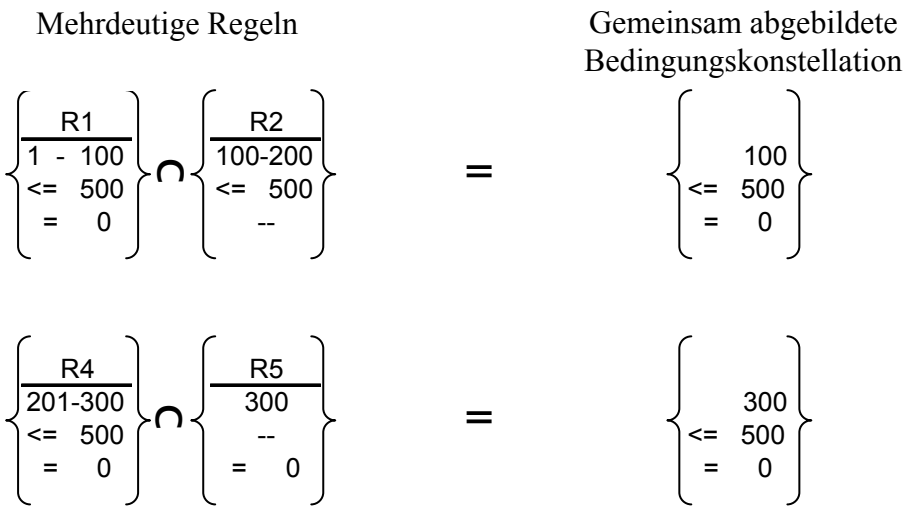
		1	2	3	4	5
B1	Schlüssel	1-100	100-200	100-200	201-300	300
B2	Feld A	<=500	<=501	>500	<=501	--
B3	Feld B	= 0	--	>0	= 0	= 0
A1	Berechne X = Z	+ 100	+ 101		- 101	- 101
A2	Gehe nach Tabelle	5	5	7	8	10

1. Schritt: Diagnostik der Mehrdeutigkeit. Jede Regel wird mit jeder anderen Regel im Bedingungsanzeigerteil verglichen. In mindestens einer Zeile müssen die Bedingungsanzeiger logisch disjunkt sein, d. h. unterschiedliche Merkmale des Entscheidungsfeldes abbilden. Andernfalls besteht Mehrdeutigkeit.

Verglichene Regeln			Disjunkte Anzeiger
R1	mit	R2	keine
R1	mit	R3	in B2
R1	mit	R4	in Bl
R1	mit	R5	in Bl
R2	mit	R3	in B2
R2	mit	R4	in Bl
R2	mit	R5	in Bl
R3	mit	R4	in Bl
R3	mit	R5	in Bl
R4	mit	R5	keine

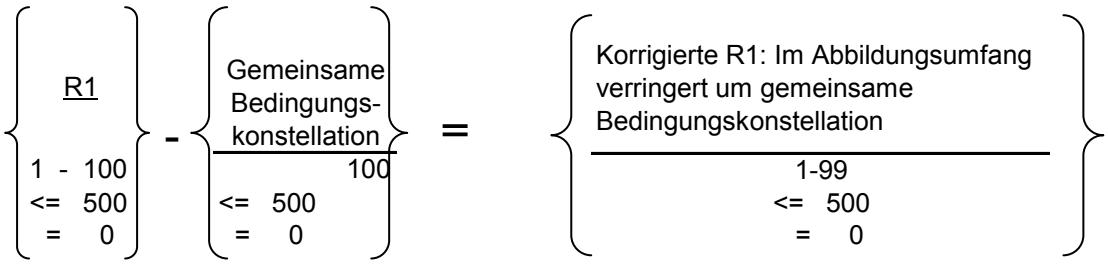
2. Schritt: Bei Regelpaaren. die in keiner Zeile logisch disjunkte Bedingungsanzeiger aufweisen, wird die gemeinsam abgebildete Bedingungskonstellation als Durchschnitt des Abbildungsumfanges beider Bedingungsanzeigerkombinationen ermittelt. Der Durchschnitt der Anzeigerkombinationen wird nach folgender Vorschrift formuliert:

- Bei Überschneidung der Anzeiger wird der gemeinsame Abbildungsbereich formuliert;
- bei Inklusion wird derjenige Anzeiger, der zum anderen Anzeiger in der Beziehung der Teilmenge steht, übernommen;
- bei Identität bleibt die Anzeigerformulierung unverändert.

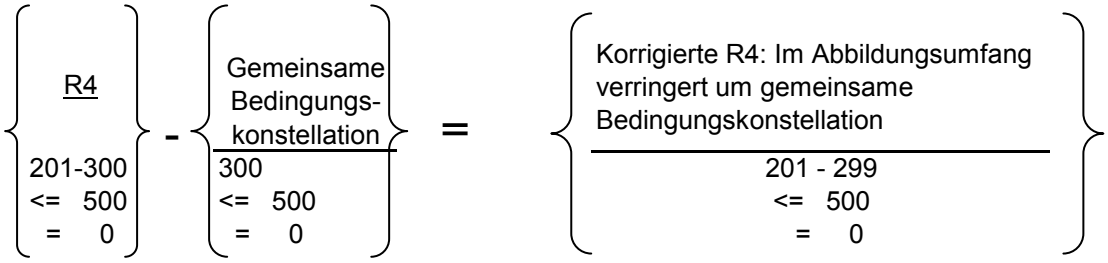


3. Schritt: Abändern jeweils einer Regel aus jedem Regelpaar, so dass die gemeinsame Bedingungskonstellation von dieser Regel nicht mehr abgebildet wird. Wenn beide Regeln identische Aktionsfolgen enthalten, ist es logisch unerheblich, aus welcher Regel die gemeinsame Bedingungskonstellation eliminiert wird. Wenn die betroffenen Regeln unterschiedliche Aktionsfolgen enthalten, besteht ein logischer Widerspruch. Welche von beiden Regeln korrigiert werden soll, kann nur entschieden werden, wenn geklärt worden ist, welche der beiden unterschiedlichen Aktionsfolgen für die gemeinsame Bedingungskonstellation gelten soll.

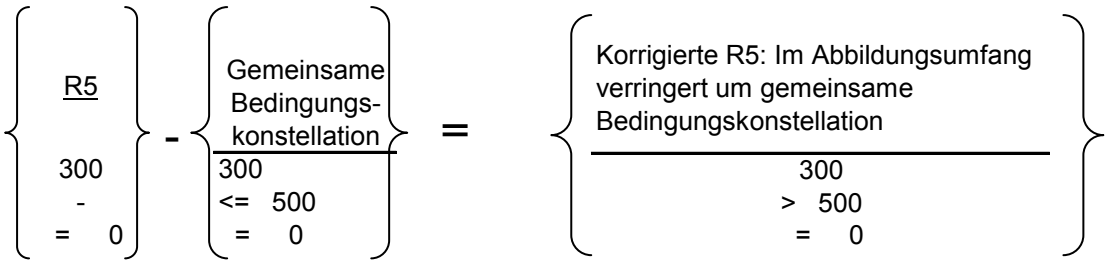
Regelpaar R1 und R2. Aktionsfolgen sind identisch.
Es wird willkürlich bestimmt, R1 zu korrigieren.



Regelpaar R4 und R5. Aktionsfolgen sind unterschiedlich.
Bei Korrektur von R4:



Bei Korrektur von R5:



1.2.2.2. Analyse semantischer Unvollständigkeit

Eine Entscheidungstabelle ist semantisch unvollständig, wenn sie nicht alle entscheidungsrelevanten Merkmale des Entscheidungsfeldes abbildet.
Verifikationsprozeduren werden benötigt, die Unvollständigkeiten in allen logischen Bereichen der Entscheidungstabelle diagnostizieren. Dabei können in einigen Fällen syntaktische Prüfungsverfahren hinzugezogen werden.

1.2.2.2.1 Ermittlung fehlender Bedingungen und Aktionen

Der methodische Nachweis der Vollständigkeit der *Bedingungen* und *Aktionen* in einer Entscheidungstabelle ist ein noch ungelöstes Problem. Syntaktische Prüfverfahren können nicht hinzugezogen werden, da sich diese nur auf die Prüfung der Vollständigkeit der Bedingungsanzeigerkombinationen erstrecken und die Vollständigkeit der Bedingungen und Aktionen voraussetzen.

Die in der Datenverarbeitungsliteratur beschriebenen *Istaufnahmetechniken*[1)], wie z. B. Interviewtechniken, Analyse vorhandener Unterlagen oder Erhebungen mittels Fragenbogen, sind natürlich zur Informationsgewinnung über relevante Bedingungen und Aktionen eines Entscheidungsfeldes geeignet, allen diesen Techniken haftet jedoch der grundlegende Mangel an, dass sie nur in Ausnahmefällen auch den Beweis erlauben, dass die ermittelten Fakten vollständig und fehlerfrei sind.

Die im Folgenden beschriebene Methode versucht, die logischen Beziehungen zwischen Bedingungen und Aktionen zur Vervollständigung von Bedingungen und Aktionen auszunutzen.

Die bereits - beispielsweise mit Hilfe von Istaufnahmetechniken - ermittelten Bedingungen und Aktionen werden zunächst matrixförmig angeordnet:

	Aktion α	Aktion β	Aktion γ	Aktion δ
Bedingung a				
Bedingung b				
Bedingung c				

Nun wird zunächst der Einfluss von Bedingung a auf die aufgeführten Aktionen untersucht. Wenn erkannt wird, dass vor Ausführung einer der Aktionen der Zustand der Bedingung a bekannt sein muss, wird dieser Einfluss durch ein "X" als Matrixelement im Schnittpunkt von Bedingung a und der beeinflussten Aktion gekennzeichnet. Angenommen, Bedingung a beeinflusst die Aktionen α und δ, dann ergibt sich nach dem ersten Analyseschritt folgende Matrix:

---

1) Vgl. dazu Pärli (Istaufnahme), S. 100 ff.

	Aktion α	Aktion β	Aktion γ	Aktion δ
Bedingung a	X			X
Bedingung b				
Bedingung c				

Nun wird zusätzlich geprüft, ob die Bedingung a eine weitere Aktion beeinflussen könnte, die noch nicht in der Matrix aufgeführt ist. In vielen Fällen lassen sich auf diese Weise zusätzliche Aktionen finden, denn gezielte Analysen der Auswirkungen einer einzigen Bedingung erbringen mehr Information über das Entscheidungsfeld als globale Analysen der Auswirkungen ganzer Bedingungsgruppen. Angenommen, Bedingung a beeinflusst eine weitere Aktion, die hier "Aktion ε" genannt werden soll. Die Matrix hat dann nach dem zweiten Analyseschritt folgendes Aussehen:

	Aktion α	Aktion β	Aktion γ	Aktion δ	Aktion ε
Bedingung a	X		X	X	X
Bedingung b					
Bedingung c					

Nachdem auch die Bedingungen b und c in gleicher Weise analysiert werden, ist der Prozess der Ermittlung zusätzlicher Aktionen zunächst abgeschlossen. Hier soll angenommen werden, dass noch drei weitere zusätzliche Aktionen ermittelt wurden, so dass sich folgende Matrix ergibt:

	Aktion α	Aktion β	Aktion γ	Aktion δ	Aktion ε	Aktion ξ	Aktion υ	Aktion ι
Bedingung a	X	X	X	X	X	X	X	X
Bedingung b		X				X		
Bedingung c	X	X	X	X			X	X

Nun wird noch ergänzend eine ähnliche Analyse spaltenweise durchgeführt. Für jede Aktion wird isoliert geprüft, ob

- die Prüfung der angekreuzten Bedingungen zur Auslösung der Aktion ausreicht

oder

- weitere, bereits in der Matrix angeführte Bedingungen einen Einfluss ausüben, der zunächst nicht erkannt wurde,

oder

- zusätzliche Bedingungen erforderlich sind.

Nach dieser Analyse könnte sich folgende Matrix ergeben:

	Aktion α	Aktion β	Aktion γ	Aktion δ	Aktion ε	Aktion ξ	Aktion υ	Aktion ι
Bedingung a	X		X	X	X			
Bedingung b		X				X		
Bedingung c	X	X	X	X			X	X
Bedingung d	X							
Bedingung e		X	X					
Bedingung f					X	X		

Auch damit ist die Analyse noch nicht beendet. Die zusätzlich ermittelten Bedingungen können wieder weitere Aktionen beeinflussen und diese wiederum die Prüfung zusätzlicher Bedingungen erfordern. Erst dann, wenn ein stabiles Ergebnis erreicht ist, wird mit dem Aufbau der Entscheidungstabelle

oder eines Entscheidungstabellenverbunds begonnen. Ausdrücklich soll noch einmal betont werden, dass auch die vorgestellte Methode keinen schlüssigen Beweis für die Vollständigkeit von Bedingungen und Aktionen liefert; sie ist jedoch praktikabel und kann mit Erfolg zusätzlich zu den bekannten Istaufnahmetechniken eingesetzt werden.

Die Grenzen der vorgestellten Methode liegen darin, dass keine fehlende Bedingung (oder Aktion) ermittelt werden kann, die nicht mit einer bereits vorhandenen Aktion (oder Bedingung) logisch verknüpft ist.

Ergänzend muss deshalb eine ablauforientierte Überprüfung der Bedingungen und Aktionen in sich erfolgen. Dazu wird je eine semantisch zulässige Basisfolge der ermittelten Bedingungen und Aktionen gebildet. Rein syntaktisch betrachtet, ist jede Anordnung von Bedingungen und Aktionen als Basisfolge geeignet. Semantisch als Basisfolge zulässig ist eine Anordnung von Bedingungen jedoch nur dann, wenn die Bedingungen in dynamischer Sicht auch in der angegebenen Reihenfolge überprüft werden können. Ebenso ist eine Anordnung von Aktionen nur dann als Basisfolge zulässig, wenn die Aktionen in der angegebenen Reihenfolge ausgeführt werden können. Logische Abhängigkeiten innerhalb der Bedingungen oder Aktionen limitieren die Anzahl semantisch zulässiger Basisfolgen oft erheblich.

Erkennbare logische Abhängigkeiten von Bedingungen oder Aktionen können jedoch auch der Ermittlung fehlender Bedingungen oder Aktionen dienen. Dazu wird der Ablauf der Bedingungsprüfungen und Aktionsausführungen simuliert und auf Plausibilität überprüft. Aus semantischer Sicht fehlende Bedingungen oder Aktionen werden in die Basisfolgen einbezogen.

1.2.2.2.2 Vollständigkeit bei der Abbildung von Bedingungszuständen und Aktionsausprägungen

Sofern in einer Entscheidungstabelle nicht ausschließlich einfache Bedingungen und Aktionen verwendet werden, so dass ausschließlich die Hilfszeichen "J", "N" und "X" Verwendung finden, besteht die Notwendigkeit der Verifikation, dass die aufgeführten Bedingungs- und Aktionsanzeiger tatsachlich alle Bedingungszustände und Aktionsausprägungen des realen Entscheidungsfeldes abbilden.

In Fällen, in denen sich unterschiedliche Bedingungszustände oder Aktionsausprägungen nach klar erkennbaren Gesetzmäßigkeiten entwickeln, kann

die Verifikation der Anzeiger häufig bereits auf syntaktischer Grundlage erfolgen. Einige Beispiele sollen diese Behauptung erläutern.

1. Für den Ablauf eines Entscheidungsprozesses ist es erforderlich, den Inhalt zweier Datenfelder miteinander zu vergleichen. Folgende Formulierungen der Bedingung und ihrer Anzeiger sind bereits auf syntaktischer Grundlage verifizierbar:

   Vergleich Feld A mit Feld B: gleich, ungleich

   Vergleich Feld A mit Feld B: größer, gleich, kleiner

   Ohne den Inhalt der beiden Datenfelder oder gar den Zweck des Vergleichs überhaupt zu kennen, können die jeweils gewählten Bedingungsanzeiger leicht als vollständige Abbilder der real möglichen Bedingungszustände erkannt werden.

2. In Abhängigkeit von der Dauer ihrer Betriebszugehörigkeit erhalten Arbeitnehmer eine Jahresprämie.
   Bei folgender Bedingungsformulierung ist ebenso bereits auf syntaktischer Grundlage verifizierbar, dass alle Bedingungszustände abgebildet werden.

   Betriebszugehörigkeit in Jahren: <=3, 4 – 9, >= 10

3. Im Rahmen eines Arbeitsplanungsproblems wird in Abhängigkeit von der Art der auszuführenden Arbeitsgänge in einer Entscheidungstabelle eine geeignete Maschine ausgewählt. Wenn bekannt ist, dass nur 8 Maschinen vorhanden sind, ist die folgende Aktion leicht verifizierbar:

   Benutzte Maschine: R10, R18, R25, R30, R36, R40, R50, R66

Die Verifikation von Bedingungs- und Aktionsanzeigern wird schwierig, wenn Bedingungszustände und Aktionsausprägungen keinerlei Gesetzmäßigkeiten unterliegen oder Elemente einer großen Liste sind, die nur mit unvertretbar hohem Aufwand vollständig erfassbar wäre.

Zumindest die vollständige Abbildung der *Bedingungszustände* lässt sich dann durch Verwendung des ELSE -Bedingungsanzeigers sicherstellen.[1)] Der Systemplaner kann einige Entscheidungsregeln seiner Entscheidungsstabelle so gestalten, dass er besondere Aktionen für den Fall vorsieht, dass ein ELSE -Bedingungsanzeiger bei der Bedingungsprüfung als zutreffend erkannt wird.

---

1) Vgl. dazu Abschnitt 1.1.1.1.1.

Die Verifikation der Anzeiger kann durch die Eigenschaften der benutzten Primärsprache begünstigt oder behindert werden. Natürliche Sprachen beinhalten einerseits einen großen Sprachumfang, andererseits jedoch auch eine Fülle von Redundanzen, die oft eine eindeutige Ermittlung des Bezeichneten nicht zulassen. So bestehen beispielsweise bei folgender Formulierung

Alter: bis 21, ab 21

durchaus Zweifel, welcher der beiden Bedingungsanzeiger das Alter = 21 einschließen soll.

Formale Sprachen dagegen sind üblicherweise so aufgebaut, dass ihre Ausdrücke leicht syntaktisch überprüfbar sind. Als Folge davon ist ihr Sprachumfang gegenüber natürlichen Sprachen erheblich eingeschränkt, und die Möglichkeit redundanter Formulierungen ist gering. Beispielsweise würde eine formale Sprache den Benutzer dazu zwingen, den Bedingungszustand "= 21" deutlich einem der beiden Anzeiger zuzuordnen. Die Bedingung könnte in COBOL so formuliert werden:

ALTER-1: LESS THAN 21, GREATER OR EQUAL TO 21

Aufgrund des eingeschränkten Sprachumfangs kann es bei Verwendung formaler Sprachen aber auch erforderlich werden, Bedingungen, Aktionen und Anzeiger als Kombination syntaktisch zulässiger Ausdrücke aufzubauen, die wiederum die Erkennung des Gemeinten erschweren.

1.2.2.2.3 Vollständigkeit bei der Abbildung der Bedingungskonstellationen

Unter der Voraussetzung, dass Bedingungen und Bedingungsanzeiger in einer Entscheidungstabelle vollständig sind, d. h. die entsprechenden Teile des Entscheidungsfeldes korrekt abbilden, lässt sich der Nachweis der vollständigen Abbildung aller möglichen Bedingungskonstellationen unter Verwendung des Instrumentariums der mathematischen Logik erbringen.

Die Bedingungen eines Entscheidungsfeldes lassen sich formal als Aussagen auffassen, so dass insbesondere die Theorie der aussagenlogischen Normalformen[1)] anwendbar wird. Jede noch so komplex aufgebaute Bedingungskonstellation der Realität muss sich durch einen entsprechenden Ausdruck in Form einer aussagenlogischen Normalform abbilden lassen.

Von den beiden Normalformen, der kanonisch konjunktiven und der kanonisch alternativen, ist nur die kanonisch alternative Normalform für Zwecke der Entscheidungstabellentechnik geeignet. Das soll an einem Beispiel verdeutlich werden.

---

1) Vgl. dazu Asser (Mathematische Logik), S. 44 ff.

Alle logischen Funktionen der

Bedingung a mit [a]= {a1, a2, a3} und
Bedingung b mit [b]= {bl, b2}

sollen durch die beiden kanonischen Normalformen repräsentiert werden.
Dann soll festgestellt werden, welche der Elementarkonjunktionen bzw. Elementaralternativen bei allen Wahrheitswertkombinationen zutreffen.

Erfüllte Bedingungs-zustände					Kanonisch konjunktive Normalform					
a1	a2	a3	b1	b2	(a1 V b1)	∧ (a1 V b2)	∧ (a2 V b1)	∧ (a2 V b2)	∧ (a3 V b1)	∧ (a3 V b2)
W	F	F	W	F	W	W	W	F	W	F
W	F	F	F	W	W	W	F	W	F	W
F	W	F	W	F	W	F	W	W	W	F
F	W	F	F	W	F	W	W	W	F	W
F	F	W	W	F	W	F	W	F	W	W
F	F	W	F	W	F	W	F	W	W	W

Erfüllte Bedingungs-zustände					Kanonisch alternative Normalform					
a1	a2	a3	b1	b2	(a1 V b1)	∧ (a1 V b2)	∧ (a2 V b1)	∧ (a2 V b2)	∧ (a3 V b1)	∧ (a3 V b2)
W	F	F	W	F	W	F	F	F	F	F
W	F	F	F	W	F	W	F	F	F	F
F	W	F	W	F	F	F	W	F	F	F
F	W	F	F	W	F	F	F	W	F	F
F	F	W	W	F	F	F	F	F	W	F
F	F	W	F	W	F	F	F	F	F	W

Daraus wird ersichtlich, dass bei der kanonisch konjunktiven Normalform bei jeder Kombination der Bedingungszustände vier unterschiedliche Elementaralternativen zutreffen. Bei der kanonisch alternativen Normalform dagegen trifft bei jeder Kombination der Bedingungszustände nur jeweils

eine Elementarkonjunktion zu, wobei es sich jedes Mal um eine andere handelt. Da in einer Entscheidungstabelle jede Regel eine andere Aktionsfolge beinhalten kann, lässt sich die Logik der Entscheidungstabelle am besten mit Hilfe der kanonisch alternativen Normalform vergleichen.

Wenn so die elementaren Bedingungsanzeigerkombinationen der Entscheidungstabelle als Analogie zu den Elementarkonjunktionen gesehen werden müssen, ergibt sich die alternative Verknüpfung der Anzeigerkombinationen untereinander ebenso logisch zwingend.

Zur Verifikation der Entscheidungstabelle ist es erforderlich nachzuweisen, dass alle Elementarkonjunktionen der Bedingungszustände in der Entscheidungstabelle berücksichtigt sind bzw. ihr Fehlen logisch zulässig ist.

Der erste Schritt einer semantischen Vollständigkeitsprüfung im Bereich der Bedingungskonstellationen kann die rechnerische syntaktische Vollständigkeitsprüfung sein. Ihre Ergebnisse sind jedoch nur dann semantisch relevant, wenn die Bedingungsanzeiger jeder Bedingung logisch disjunkt sind.

Die Definition des Indifferenzanzeigers "_" als Symbol für die Menge aller Bedingungsanzeiger der betreffenden Bedingung dagegen kann auch aus semantischer Sicht bestätigt werden. Wenn eine Bedingung innerhalb einer Bedingungskonstellation der Realität keinen Einfluss auf die zugeordnete Aktionsfolge ausübt, dann kann entweder auf ihre Überprüfung verzichtet werden, oder sie muss - wenn sie aus bestimmten Gründen aufgeführt werden soll - dadurch in ihren Auswirkungen logisch neutralisiert werden, dass alle zulässigen Bedingungszustände erkennbar zur gleichen Aktionsfolge führen. Danach ist die semantische Bedeutung folgender Entscheidungstabellen gleich:

Prämisse: Bedingung c mit [c]= {c1, c2, c3}

		1	2
B1	Bedingung a	J	N
B2	Bedingung b	J	N
A1	Aktion α	X	
A2	Aktion β		X

		1	2	3	4	5	6
B1	Bedingung a	J	J	J	N	N	N
B2	Bedingung b	J	J	J	N	N	N
B3	Bedingung c	c1	c2	c3	c1	c2	c3
A1	Aktion α	X	X	X			
A2	Aktion β				X	X	X

Semantisch äquivalent wäre schließlich auch die Form:

		1	2
B1	Bedingung a	J	N
B2	Bedingung b	J	N
B3	Bedingung c	-	-
A1	Aktion α	X	
A2	Aktion β		X

Auf einen weiteren entscheidenden Unterschied zwischen syntaktischer und semantischer Vollständigkeit im Bereich der Bedingungskonstellationen muss noch hingewiesen werden. Bei der syntaktischen Vollständigkeitsprüfung wird von der semantischen Bedeutung der Bedingungen völlig abstrahiert und eine unbeschränkte Kombinierbarkeit der Bedingungsanzeiger vorausgesetzt. Die semantische Vollständigkeitsprüfung kann diese Einschränkung nicht mehr aufrechterhalten: Die gegenseitigen logischen Einflussnahmen der Bedingungen innerhalb der Entscheidungstabelle müssen sorgfältig analysiert und bei der semantischen Vollständigkeitsprüfung in ihren Auswirkungen berücksichtigt werden.

In der Entscheidungstabellenliteratur hat King [1)] dieses Problem zum ersten Mal pointiert herausgearbeitet. Er zeigt am Beispiel folgender Trivialtabelle

---

1) King (Ambiguity), S. 680

		1	2	3
B1	Alter < 18	J	-	N
B2	Alter > 65	-	J	N
A1	Gehe nach	1	2	3

dass aufgrund logischer Abhängigkeit der Bedingungen in einer Entscheidungstabelle durchaus Differenzen zwischen syntaktischer Vollständigkeit und semantischer Vollständigkeit auftreten können.

Unter weitgehender Berücksichtigung der syntaktischen Vollständigkeitsprüfung, aber auch in klarer Kenntnis ihrer Grenzen, kann eine semantische Vollständigkeitsprüfung in folgenden Teilschritten erfolgen.

1. Prüfung, ob die Bedingungsanzeiger jeder Bedingung alle semantisch relevanten Bedingungszustände abbilden und logisch disjunkt sind.

2. Wenn die unter 1. angegebene Prüfung positiv verläuft, wird eine syntaktische Vollständigkeitsprüfung durchgeführt. Ergibt diese Prüfung, dass alle syntaktisch möglichen Bedingungsanzeigerkombinationen berücksichtigt sind, dann kann das gleichzeitig als Nachweis der semantischen Vollständigkeit gelten, und die Prüfung ist mit positivem Ergebnis beendet.

3. Wenn die Tabelle bei der syntaktischen Vollständigkeitsprüfung als unvollständig erkannt wurde, werden - ebenfalls auf syntaktischer Grundlage - die fehlenden Bedingungsanzeigerkombinationen ermittelt.

4. Die semantische Relevanz der fehlenden Bedingungsanzeigerkombinationen ist zu prüfen, d. h. es ist festzustellen, ob die fehlenden Bedingungsanzeigerkombinationen Abbildungen real möglicher Bedingungskonstellationen sind. Dazu wird in der Regel die Prüfung erforderlich sein, ob die Bedingungen logisch abhängig sind.

5. Auf Bedingungsanzeigerkombinationen, die logisch unzulässige Bedingungskonstellationen beschreiben, kann, ohne die semantische Vollständigkeit der Entscheidungstabelle zu beeinträchtigen, verzichtet werden. Fehlende Bedingungsanzeigerkombinationen, die logisch zulässige Bedingungskonstellationen beschreiben, müssen jedoch in vorhandene Regeln einbezogen werden oder zur Ergänzung weiterer Regeln führen.

Das folgende Beispiel soll den Ablauf der semantischen Vollständigkeitsprüfung illustrieren. Geprüft werden soll folgende Tabelle:

Auftragsbestätigung		1	2	3	4	5	6	7	8	9	10
B1	Artikelgruppe	A	A	A	A	A	A	B	B	B	B
B2	Gewünsche Liefer-zeit in Tagen	<14	<14	14-30	14-30	14-30	>30	<=30	<=30	<=30	>30
B3	Lagerbestand % bestellte Menge >= Mindestbestand	J	N	J	N	N	-	J	N	N	-
B4	Mindestbestand würde durch Auftragsausführung um mehr als 1/3 unterschritten	N	-	N	J	N	-	N	J	N	-
A1	Auftrag bestätigen	X		X		X	X	X			X
A2	Auftrag unter Vorbehalt einer längeren Lieferzeit bestätigen				X					X	
A3	Mögliche Lieferzeit dem Kunden tele-fonisch mitteilen und seine Entschei-dung einholen		X						X		

1. Schritt: Prüfung, ob die Bedingungsanzeiger alle Bedingungszustände abbilden und disjunkt sind

Unter der Voraussetzung, dass nur die Artikelgruppen A und B existieren, sind die Bedingungsanzeiger vollständig. Bei B2 überlappen sich die Bedingungsanzeiger teilweise. Durch Splitten des Bedingungsanzeigers "<= 30" in den Regeln 7, 8 und 9 in "<14" und "14 - 30" wird die Überlappung beseitigt. Es entstehen dadurch drei zusätzliche Regeln.

Auftragsbestätigung		1	2	3	4	5	6	7	8	9	10	11	12	13
B1	Artikelgruppe	A	A	A	A	A	A	B	B	B	B	B	B	B
B2	Lieferzeit	<14	<14	14-30	14-30	14-30	>30	<14	14-30	<14	14-30	<14	14-30	>30
B3	Lagerbestand...	J	N	J	N	N	-	J	J	N	N	N	N	-
B4	Unterschreitung	N	-	N	J	N	-	N	N	J	J	N	N	-
A1	Bestätigung	X		X		X	X	X	X					X
A2	Vorbehalt				X							X	X	
A3	Telefonische Mitteilung		X							X	X			

2. Schritt: Syntaktische Vollständigkeitsprüfung

Formal sind maximal 2 . 3 . 2 . 2 = 24 Bedingungsanzeigerkombinationen möglich. Mit den vorhandenen Regeln der Tabelle sind (von links nach rechts)

1 + 2 + 1 + 1 + 1 + 4 + 1 + 1 + 1 + 1 + 1 + 1 + 4 = 20

Bedingungsanzeigerkombinationen abgedeckt. Es fehlen also 4 Bedingungsanzeigerkombinationen.

3. Schritt: Ermitteln der fehlenden Bedingungsanzeigerkombinationen
Syntaktisch können folgende fehlenden Bedingungsanzeigerkombinationen ermittelt werden:

A	A	B	B
<14	14-30	<14	14-30
J	J	J	J
J	J	J	J

4. Schritt: Prüfen der semantischen Relevanz
Alle fehlenden Bedingungsanzeigerkombinationen sind semantisch nicht relevant, da sie unlogische Fälle bezeichnen.

Wenn die bestellte Menge nicht zur Unterschreitung der Mindestmenge führt, kann der Mindestbestand auch nicht um mehr als 1/3 unterschritten werden.

5. Schritt: Ergänzung der Tabelle

In diesem Fall nicht erforderlich, die Tabelle ist semantisch vollständig.

Auch bei logischer Abhängigkeit der Bedingungen wäre die Vollständigkeitsprüfung auf Syntaxbasis möglich (und damit automatisierbar), wenn einem Vorschlag von Pollack, Hicks und Harrison zur Verwendung besonderer Bedingungsanzeiger gefolgt würde.

"*" in einer Bedingungsanzeigerkombination soll anzeigen, dass die betreffende Bedingung innerhalb der beschriebenen Konstellation aufgrund logischer Abhängigkeiten nur *nicht erfüllt* sein kann.

"$" soll anzeigen, dass die betreffende Bedingung innerhalb der beschriebenen Konstellation aufgrund logischer Abhängigkeiten nur *erfüllt* sein kann.[1)]

Unabhängig von ihrem semantischen Inhalt können zum Zweck der Vollständigkeitsprüfung alle "*" - und "$" -Anzeiger als Indifferenzanzeiger angesehen werden, so dass die Auswirkung der logischen Abhängigkeit von Bedingungen neutralisiert wird.[2)]

Dieser Vorschlag kann jedoch eine Lösung des angesprochenen Problems nur für begrenzte Entscheidungstabellen bringen. Bei Verwendung komplexer Bedingungen lässt sich eine ähnliche Wirkung durch Einführung zusätzlicher Anzeiger mit besonderen Funktionen nicht erzielen, da die Anzahl der abgebildeten Bedingungszustände größer als zwei und variabel ist. Es erscheint jedoch bedenklich, irgendeine Spracherweiterung einzuführen die nur bei begrenzten Entscheidungstabellen anwendbar ist. Der Vorschlag von Pollack, Hicks und Harrison kann deshalb hier nicht unterstützt werden.

Leistungsfähiger, da bei jeder Tabellenart anwendbar, ist ein Vorschlag der Decision Tables Task Group der CODASYL. Danach soll ein Bedingungsanzeiger innerhalb einer Kombination den Zusatz "!" erhalten, wenn er aufgrund semantischer Gegebenheiten als einziger der syntaktisch definierten Liste von Bedingungsanzeigern einer Bedingung zutreffen kann.[3)]

---

1) Vgl. Pollack; Hicks; Harrison (Decision tables), S. 30

2) Vgl. Donnell (Decision tables checks), S. 5 ff. und Harrison (Complete decision tables), S. 90 ff.

3) Vgl. Theory subgroup (DT glossary), S. 8 f.
Vgl. dazu auch Verhelst (Conversion of limited-entry decision tables) S. 975, der den Zusatz "*" zur Kennzeichnung logischer Abhängigkeiten vorschlägt.

Ein mit "!" versehener Bedingungsanzeiger kann im Rahmen der syntaktischen Vollständigkeitsprüfung wie ein Indifferenzanzeiger behandelt werden.

1.2.2.2.4 Vollständigkeit bei der Abbildung der Aktionsfolgen

Die innerhalb der Regeln einer Entscheidungstabelle enthaltenen Aktionsanzeigerkombinationen sind stets eine Teilmenge der insgesamt möglichen Aktionsanzeigerkombinationen. Die prinzipiell gegebene syntaktische Möglichkeit - ähnlich wie bei der Vollständigkeitsprüfung im Bereich der Bedingungskonstellationen -, einen Vergleich der aufgeführten Anzeigerkombinationen mit den formal möglichen Anzeigerkombinationen durchzuführen, ist semantisch bedeutungslos.

Lediglich eventuelle logische Abhängigkeiten der Aktionen können die semantische Kontrolle der Vollständigkeit der Abbildung der Aktionsfolgen erleichtern. Kann beispielsweise eine Aktion nur in Kombination mit einer anderen Aktion logisch sinnvoll ausgeführt werden, dann lässt sich prüfen, ob alle aufgeführten Aktionsanzeigerkombinationen, die die eine Aktion enthalten, auch die andere Aktion berücksichtigen. Darüber hinaus verbleibt nur noch die Möglichkeit, jede der aufgeführten Aktionsfolgen in sich auf logische Plausibilität zu prüfen und zusätzlich festzustellen, ob alle der Basisfolge angehörenden Aktionen mit allen definierten Aktionsausprägungen in mindestens einer der aufgeführten Aktionsanzeigerkombinationen syntaktisch berücksichtigt wurden.

1.2.3 Minimierung der Entscheidungstabelle

Eine Entscheidungstabelle gilt als minimiert, wenn keine alternative Entscheidungstabelle mehr gefunden werden kann, die ein identisches Entscheidungsfeld mit weniger syntaktischen Einheiten abbildet. Obwohl in der Entscheidungstabellen-Literatur bisher nicht allgemein gebräuchlich, wird die Bezeichnung "Minimierung" für alle syntaktischen und semantischen Transformationen verwendet, deren Zielsetzung die Verringerung der syntaktischen Einheiten einer Entscheidungstabelle unter Beibehaltung ihrer Abbildungslogik ist. Das geschieht aufgrund der formalen Analogie zu den Verfahren der Minimierung von Schaltkreisen.

Auslösender Faktor für die Minimierung einer Entscheidungstabelle sind rein pragmatische Gesichtspunkte. Der Anwender einer Entscheidungstabelle - sei dies nun ein personaler oder ein maschineller Aktionsträger - erfasst die Logik einer Entscheidungstabelle umso leichter, je weniger syn-

taktische Einheiten diese beinhaltet. Die Minimierung der syntaktischen Einheiten einer Entscheidungstabelle hat in der Regel auch eine Minimierung des Entscheidungsprozesses zur Folge, der auf der Basis dieser Entscheidungstabelle durchgeführt wird.

Besondere Beachtung verdient jedoch die Minimierung des *Entscheidungsstabellenverbunds*. Wie zu zeigen sein wird, resultiert ein minimaler Entscheidungstabellenverbund nicht notwendigerweise aus der Minimierung aller enthaltener Einzeltabellen.

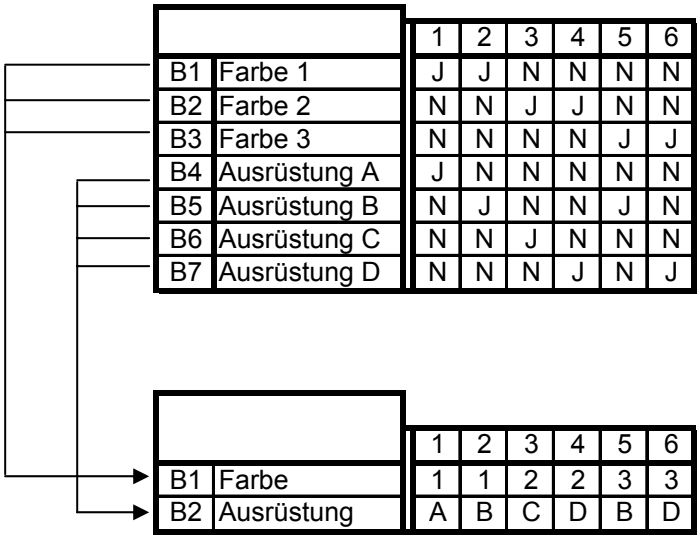
1.2.3.1 Minimierung von Bedingungen und Aktionen

Eine Möglichkeit der Reduzierung der Bedingungen bzw. Aktionen einer Entscheidungstabelle ohne Veränderung ihres Abbildungsumfangs ist die Umwandlung einfacher Bedingungen (Aktionen) in komplexe Bedingungen (Aktionen).

Lässt sich in einer Entscheidungstabelle eine Gruppe von einfachen Bedingungen ermitteln, die derart voneinander logisch abhängig sind, dass immer dann, wenn eine Bedingung der Gruppe erfüllt ist, die anderen Bedingungen nicht erfüllt sind, dann ist in der Regel eine Zusammenfassung der ganzen Bedingungsgruppe zu einer einzigen komplexen Bedingung möglich.

Dabei wird die komplexe Bedingung syntaktisch so gebildet, dass der allen Bedingungen gemeinsame Bedingungstext als neue Bedingung gilt und die bei allen Bedingungen unterschiedlichen Textteile als Anzeiger der komplexen Bedingung definiert werden.

Das folgende Beispiel illustriert die Umformung einfacher Bedingungen in komplexe Bedingungen.

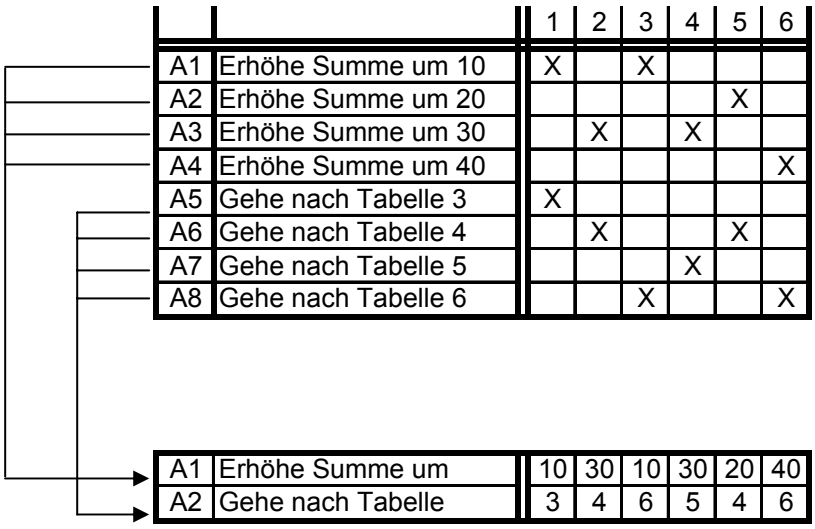


		1	2	3	4	5	6
B1	Farbe 1	J	J	N	N	N	N
B2	Farbe 2	N	N	J	J	N	N
B3	Farbe 3	N	N	N	N	J	J
B4	Ausrüstung A	J	N	N	N	N	N
B5	Ausrüstung B	N	J	N	N	J	N
B6	Ausrüstung C	N	N	J	N	N	N
B7	Ausrüstung D	N	N	N	J	N	J

		1	2	3	4	5	6
B1	Farbe	1	1	2	2	3	3
B2	Ausrüstung	A	B	C	D	B	D

Eine ähnliche Transformation verwandelt einfache Aktionen in komplexe Aktionen.

Beispiel:



		1	2	3	4	5	6
A1	Erhöhe Summe um 10	X		X			
A2	Erhöhe Summe um 20					X	
A3	Erhöhe Summe um 30		X		X		
A4	Erhöhe Summe um 40						X
A5	Gehe nach Tabelle 3	X					
A6	Gehe nach Tabelle 4		X			X	
A7	Gehe nach Tabelle 5				X		
A8	Gehe nach Tabelle 6			X			X

		10	30	10	30	20	40
A1	Erhöhe Summe um						
A2	Gehe nach Tabelle	3	4	6	5	4	6

Eine Reduzierung der Bedingungen kann manchmal auch dadurch erreicht werden, dass semantisch irrelevante Bedingungen aus der Tabelle entfernt werden. Eine Bedingung ist dann semantisch irrelevant, wenn ihre Überprüfung die Auswahl der in der Tabelle abgebildeten Aktionsfolgen nicht beeinflusst. Das ist beispielsweise dann der Fall, wenn die Zustände einer Bedingung an die Bedingungszustände einer anderen, ebenfalls in der Tabelle enthaltenen Bedingung starr gekoppelt sind. In diesen Fällen ist es ausreichend, die Primärbedingung in der Tabelle zu belassen. Im folgenden Beispiel ist die Bedingung b starr an Bedingung a und die Bedingung e starr an Bedingung d gekoppelt. Die Bedingungen b und e können deshalb entfernt werden, ohne dass sich die abgebildete Entscheidungslogik inhaltlich verändert.

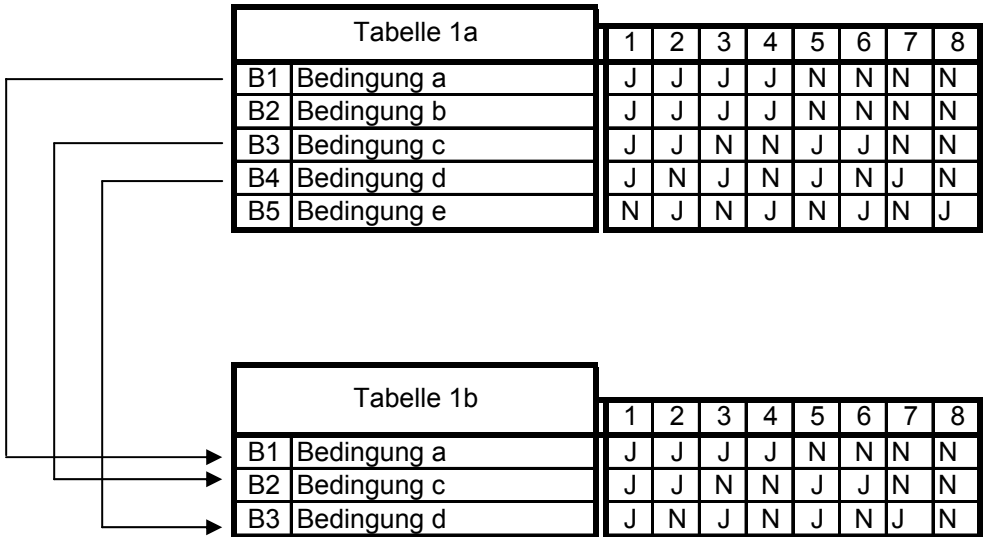
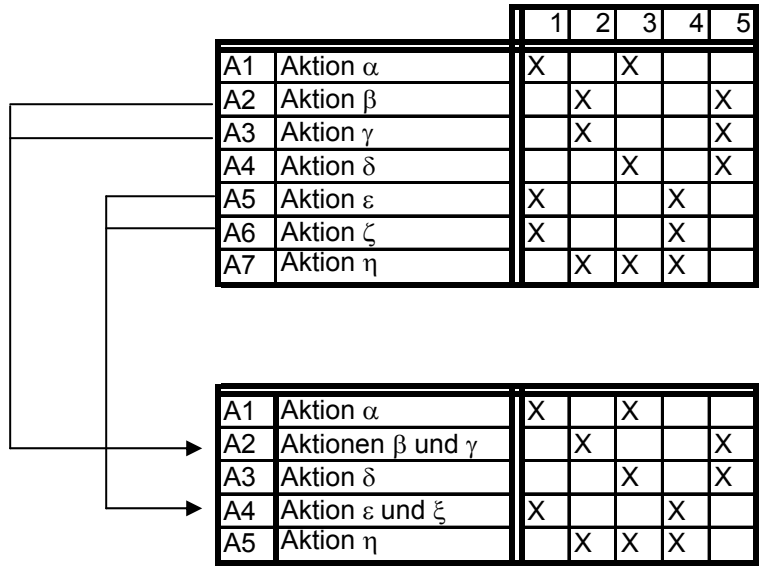


Tabelle 1a		1	2	3	4	5	6	7	8
B1	Bedingung a	J	J	J	J	N	N	N	N
B2	Bedingung b	J	J	J	J	N	N	N	N
B3	Bedingung c	J	J	N	N	J	J	N	N
B4	Bedingung d	J	N	J	N	J	N	J	N
B5	Bedingung e	N	J	N	J	N	J	N	J

Tabelle 1b		1	2	3	4	5	6	7	8
B1	Bedingung a	J	J	J	J	N	N	N	N
B2	Bedingung c	J	J	N	N	J	J	N	N
B3	Bedingung d	J	N	J	N	J	N	J	N

Auch Aktionen, die in einem derartigen logischen Zusammenhang stehen, dass sie immer gemeinsam ausgeführt werden müssen, können kombiniert werden. Syntaktische Voraussetzung ist jedoch, dass nur eine der zu kombinierenden Aktionen eine komplexe Aktion ist.

Beispiel:



		1	2	3	4	5
A1	Aktion α	X		X		
A2	Aktion β		X			X
A3	Aktion γ		X			X
A4	Aktion δ			X		X
A5	Aktion ε	X			X	
A6	Aktion ζ	X			X	
A7	Aktion η		X	X	X	

A1	Aktion α	X		X		
A2	Aktionen β und γ		X			X
A3	Aktion δ			X		X
A4	Aktion ε und ξ	X			X	
A5	Aktion η		X	X	X	

1.2.3.2 Minimierung von Bedingungsanzeigern

Die Anzahl der Bedingungsanzeiger einer komplexen Bedingung beeinflusst die Anzahl syntaktisch möglicher Bedingungsanzeigerkombinationen in erheblichem Umfang. Wenn es gelingt, einige der Bedingungsanzeiger einer komplexen Bedingung zu kombinieren, dann hat dies auch eine Reduzierung der Entscheidungsregeln zur Folge.

Bedingungsanzeiger lassen sich nur dann reduzieren, wenn die Anzahl semantisch relevanter Bedingungszustände geringer ist als die Anzahl der syntaktisch formulierten Bedingungsanzeiger. Erkennbar werden solche Fälle beispielsweise daran, dass Regelgruppen in der Entscheidungstabelle enthalten sind, die sich nur in den Bedingungsanzeigern einer Bedingungszeile voneinander unterscheiden, die unterschiedlichen Anzeiger aber nur eine Teilmenge der insgesamt zu dieser Bedingung definierten Bedingungsanzeiger sind.

Die Formulierung eines "kombinierten" Bedingungsanzeigers kann erfolgen, indem die Teilmenge der ursprünglichen Anzeiger durch einen gemeinsamen Ausdruck ersetzt wird, oder die ursprünglichen Anzeiger erhalten bleiben und die Anzeiger der Teilmenge lediglich durch ein ODER-Symbol zusammengefasst werden.

Auch der ELSE -Bedingungsanzeiger kann zur Kombination von Bedingungsanzeigern benutzt werden. Bei Verwendung des ELSE -Bedingungsanzeigers ist jedoch zu beachten, dass er grundsätzlich alle Bedingungszustände beschreibt, die nicht durch die anderen vorhandenen Bedingungsanzeiger definiert sind. Die Kombination von Bedingungsanzeigern mit Hilfe des ELSE -Anzeigers setzt also voraus, dass die Vollständigkeit der Bedingungsanzeiger verifiziert wurde.

1.2.3.3 Minimierung von Entscheidungsregeln

Enthalten einige Entscheidungsregeln einer Entscheidungstabelle die gleiche Aktionsfolge, dann lassen sich diese Regeln zu einer einzigen komplexen Regel zusammenfassen, wenn sich eine oder mehrere der als entscheidungsrelevant ausgewiesenen Bedingungen innerhalb dieser Regelgruppe in Wirklichkeit in Bezug auf die Auslösung der beschriebenen Aktionsfolge indifferent verhalten. Der Nachweis der Erfüllung dieser Minimierungsvoraussetzung lässt sich am sichersten über eine auf die Regelgruppe beschränkte semantische Vollständigkeitsprüfung der Bedingungskonstellationen führen, wobei nur diejenigen Bedingungen dieser Prüfung unterzogen werden, deren Indifferenz vermutet wird. Syntaktisch sind diese Bedingungen im Normalfall daran erkennbar, dass alle Bedingungsanzeiger explizit oder implizit über den Indifferenzanzeiger innerhalb der untersuchten Regelgruppe sichtbar werden, bei logischer Abhängigkeit der Bedingungen können jedoch syntaktische Anzeigerkombinationen fehlen, ohne dass die semantische Vollständigkeit beeinträchtigt wird. Das Minimierungsverfahren wird in folgenden Teilschritten durchgeführt:

1. Ermitteln einer Regelgruppe mit identischer Aktionsfolge.

2. Eliminieren aller Bedingungszeilen, die identische Bedingungsanzeiger aufweisen.

3. Die verbliebenen Bedingungsanzeigerkombinationen werden wie der Bereich der Bedingungskonstellationen einer eigenen Entscheidungstabelle behandelt und einer semantischen Vollständigkeitsprüfung unterzogen.

4. Wenn die Vollständigkeitsprüfung zu einem positiven Ergebnis führt, dann ist damit der Beweis erbracht, dass sich die bei der Prüfung berücksichtigten Bedingungen innerhalb der untersuchten Regelgruppe indifferent verhalten. Führt die Vollständigkeitsprüfung dagegen zu einem negativen Ergebnis, dann ist der Beweis erbracht, dass mindestens eine der Bedingungen noch einen Einfluss ausübt.

5. Bei positivem Ergebnis der Vollständigkeitsprüfung wird die Regelgruppe in eine einzige komplexe Entscheidungsregel transformiert, wobei alle Bedingungen ohne Einfluss den Indifferenzanzeiger erhalten.

6. Bei negativem Ergebnis der Vollständigkeitsprüfung wird die Regelgruppe so verringert, dass eine weitere Bedingung innerhalb der Gruppe identische Bedingungsanzeiger aufweist. Diese Bedingung wird zusätzlich eliminiert und wiederum eine Vollständigkeitsprüfung durchgeführt.

Das beschriebene Minimierungsverfahren soll nun an einem Beispiel illustriert werden.

Prämissen:

Bedingung a mit [a] = {a1,a2,a3}
Bedingung b mit [b] = {b1,b2}
Bedingung c mit [c] = {cl,c2,c3,c4}
Bedingung d mit [d] = {d1,d2,d3}

Alle Bedingungsanzeiger sind logisch disjunkt. Die Anzeiger jeder Bedingung bilden alle Bedingungszustände ab.

Aufgrund logischer Abhängigkeiten der Bedingungen können niemals Bedingungskonstellationen mit folgender Abbildung auftreten:

Bedingung a mit [a] = {a2}
Bedingung b mit [b] = {b2}
Bedingung d mit [d] = {d3}

		1	2	3	4	5	6	7	8	9	10	11	12
B1	Bedingung a	a1	a1	a1	a1	a2	a2	a2	a2	a2	a2	a3	a3
B2	Bedingung b	b1	b2	b2	b2	b1	b1	b1	b1	b2	b2	b1	b2
B3	Bedingung c	-	-	-	-	c1	c2	c3	c4	-	-	-	-
B4	Bedingung d	-	d1	d2	d3	-	-	-	-	d1	d2	-	-
A1	Aktion α	X						X	X			X	
A2	Aktion β		X	X	X		X	X		X	X	X	X
A3	Aktion γ		X	X	X	X				X	X	X	X

1. Schritt: Ermitteln einer Regelgruppe mit identischen Aktionsfolgen

	2	3	4	9	10	12
B1	a1	a1	a1	a2	a2	a3
B2	b2	b2	b2	b2	b2	b2
B3	-	-	-	-	-	-
B4	d1	d2	d3	d1	d2	-
A1						
A2	X	X	X	X	X	X
A3	X	X	X	X	X	X

2. Schritt: Eliminieren von Bedingungszeilen mit identischen Bedingungsanzeigern

B1	a1	a1	a1	a2	a2	a3
B4	d1	d2	d3	d1	d2	-
A1						
A2	X	X	X	X	X	X
A3	X	X	X	X	X	X

3. Schritt: Vollständigkeitsprüfung

Die syntaktische Vollständigkeitsprüfung ergibt Unvollständigkeit, da die Bedingungsanzeigerkombination fehlt.

$$\begin{Bmatrix} a2 \\ d3 \end{Bmatrix}$$

Bezogen auf die ursprüngliche Entscheidungstabelle bedeutet das das syntaktische Fehlen der Anzeigerkombination

$$\begin{Bmatrix} a2 \\ b2 \\ - \\ d3 \end{Bmatrix}.$$

Semantisch ist diese Unvollständigkeit jedoch unerheblich, da sie aufgrund logischer Abhängigkeit der Bedingungen als irrelevant gekennzeichnet wurde. Die in den ersten beiden Schritten gebildete Teiltabelle kann deshalb als vollständig gelten.

4. Schritt: Beweis der Indifferenz von Bedingungen

Die Bedingungen Bl und B4 verhalten sich innerhalb der untersuchten Regelgruppe indifferent.

4. Schritt: Transformation in komplexe Entscheidungsregel

Die Regeln 2, 3, 4, 9, 10 und 12 lassen sich zur komplexen Entscheidungsregel zusammenfassen:

B1	-
B2	b2
B3	-
B4	-
A1	
A2	X
A3	X

Nachdem die Beispieltabelle entsprechend geändert wurde, lässt sich als Nebenergebnis erkennen, dass die Bedingung d redundant ist.

		1	2	3	4	5	6	7
B1	Bedingung a	a1	-	a2	a2	a2	a2	a3
B2	Bedingung b	b1	b2	b1	b1	b1	b1	b1
B3	Bedingung c	-	-	c1	c2	c3	c4	-
B4	Bedingung d	-	-	-	-	-	-	-
A1	Aktion α	X				X	X	X
A2	Aktion β		X		X	X		X
A3	Aktion γ		X	X				X

Das dargestellte Verfahren eignet sich besonders gut zur manuellen Minimierung von Entscheidungsregeln. Ein Algorithmus zur automatischen Minimierung lässt sich eher auf der Basis des bereits erläuterten syntaktischen Minimierungsverfahrens entwickeln. [1)] Semantisch relevant wären die nach diesem Verfahren erzielten Ergebnisse jedoch nur dann, wenn

- die Bedingungsanzeiger jeder Bedingung alle relevanten Bedingungszustände abbilden,

- die abgebildeten Zustände jeder Bedingung logisch disjunkt sind,

- keine semantischen Mehrdeutigkeiten vorhanden sind.

Eine weitere Möglichkeit der Reduzierung von Entscheidungsregeln bietet die Anwendung der ELSE -Regel. Sie bildet definitionsgemäß alle Bedingungskonstellationen ab, die nicht durch die anderen Entscheidungsregeln abgedeckt werden. Die Anwendung der ELSE -Regel setzt nicht voraus, dass die durch sie abgebildeten Bedingungskonstellationen außer einer identischen Aktionsfolge bestimmte strukturelle Gesetzmäßigkeiten aufweisen, denn von den Zuständen der Bedingungen ist sie nur indirekt abhängig. Direkt abhängig ist sie nur vom Abbildungsumfang der anderen Entscheidungsregeln und von den bestehenden logischen Interdependenzen der Bedingungen. Eine unvermeidliche Auswirkung der Anwendung der ELSE -Regel ist, dass sie jede Entscheidungstabelle in Bezug auf die Abbildung der Bedingungskonstellationen vervollständigt. Diese im Grunde beabsichtigte Wir-

---

1) Vgl. dazu Abschnitt 1.1.2.1.2.1.2

kung hat jedoch auch eine Kehrseite: Auch Bedingungskonstellationen, deren semantischer Relevanz sich der Entscheidungstabellen-Entwickler nicht bewusst ist und die er deshalb auch nicht explizit abgebildet hat, sind automatisch in den Abbildungsumfang der ELSE -Regel einbezogen und erhalten möglicherweise völlig falsche Aktionen zugeordnet. Dieser Gefahr kann dadurch begegnet werden, dass im Rahmen einer semantischen Vollständigkeitsprüfung der Abbildungsumfang der ELSE -Regel genau bestimmt wird.

1.2.3.4 Minimierung des Entscheidungstabellenverbunds

Im Vergleich zur Minimierung einer einzelnen Entscheidungstabelle müssen bei der Minimierung eines Entscheidungstabellenverbunds einige Besonderheiten berücksichtigt werden.

Die Zielsetzung der Minimierung der syntaktischen Einheiten des gesamten Entscheidungstabellenverbunds kann durchaus im Gegensatz zur Zielsetzung der Minimierung des auf der Basis des Entscheidungstabellenverbunds durchzuführenden Entscheidungsprozesses stehen. So kann eine *aufbauorientierte* Minimierung des Verbunds eine Verlängerung des Entscheidungsprozesses zur Folge haben und eine *ablauforientierte* Minimierung eine Vermehrung der syntaktischen Einheiten des Verbunds.

Verständlich wird diese Behauptung erst, wenn akzeptiert wird, dass die Minimierung eines Entscheidungstabellenverbunds nicht mit der Minimierung aller enthaltenen Einzeltabellen identisch ist. Vielmehr steht die Aufteilung der Entscheidungslogik auf die einzelnen Tabellen erneut zur Disposition. Tabellen können zergliedert oder kombiniert werden, um den Entscheidungstabellenverbund zielgerecht zu transformieren. Durch die Zergliederung einer Entscheidungstabelle nach dem Verfahren der kombinierten Bedingungs-/Aktionsaufteilung [1)] kann sich beispielsweise eine Verringerung der syntaktischen Einheiten bei gleichzeitiger Verlängerung des Entscheidungsprozesses ergeben, wenn eine Bedingung aufgrund der bestehenden Beziehungen zu den Aktionen in mehreren entstehenden Teiltabellen aufgeführt werden muss. Die Dauer des Entscheidungsprozesses verlängert sich dann durch die mehrfache Überprüfung der gleichen Bedingung.

Eine weitere Besonderheit bei der Minimierung eines Entscheidungstabellenverbunds ergibt sich daraus, dass die jeweilige Zielsetzung nicht ohne Beachtung bestimmter Nebenbedingungen verfolgt werden kann. In der Regel

---

1) Vgl. dazu Abschnitt 1.1.2.2.1

werden Entscheidungstabellen nicht so stark zergliedert, dass die entstehenden Teiltabellen nur noch jeweils eine Bedingung enthalten. Auch wenn das im Hinblick auf die angestrebte Zielsetzung optimal und bei dem beschriebenen Problem semantisch durchführbar erschiene, ginge dadurch die Übersichtlichkeit des Entscheidungstabellenverbunds verloren.

Ebenso können Entscheidungstabellen eines Verbunds nicht unbegrenzt kombiniert werden; bei Entscheidungstabellen mit mehr als etwa 30 Regeln gilt die Regelermittlung als zu aufwendig.

Weder die unteren noch die oberen Grenzen der Tabellengröße sind jedoch starr fixiert. Pragmatische Einflüsse, wie beispielsweise die Art des Entscheidungstabellenanwenders - maschineller oder personeller Aktionsträger -, bestimmen die Nebenbedingungen der Minimierung.

1.2.4 Erstellung der Entscheidungstabelle

Die Erstellung von Entscheidungstabellen bedeutet, entscheidungsrelevante Bestandteile eines antizipierten Entscheidungsfeldes zu formalisieren. Die besondere Problematik dieses Vorgangs liegt darin, dass in der Regel zu Beginn der Tabellenerstellung über das abzubildende Entscheidungsfeld nur unvollkommene Information besteht. Systematische Verfahren der Tabellenerstellung sowie die erläuterten Verifikationsprozeduren können jedoch dazu beitragen, noch während der Tabellenerstellung die Problemkenntnis zu vertiefen und die Kenntnis des Entscheidungsfeldes zu vervollständigen. So ist die Tabellenerstellung in der Regel ein iterativer Prozess: Vorgänge der Abbildung bekannter entscheidungsrelevanter Bestandteile des Entscheidungsfeldes wechseln sich ab mit Verifikationsprozeduren und - dadurch impliziert - Veränderungen des perzipierten Entscheidungsfeldes und Korrekturen in den abgebildeten Bestandteilen des Entscheidungsfeldes.

Nach London [1] lassen sich zwei verschiedene systematische Verfahren der Tabellenerstellung unterscheiden: das klassische Verfahren und die Tabellenerstellung durch fortschreitende Regelentwicklung. Sonstige bekannte Verfahren der Tabellenerstellung können als Varianten dieser beiden Grundverfahren aufgefasst werden.[2]

---

1) London (Decision tables), S. 42 ff.
2) London (Decision tables), S. 75

1.2.4.1 Das klassische Verfahren zur Erstellung von Entscheidungsstabellen

In der Entscheidungstabellen-Literatur hundertfach beschrieben, kann das klassische Verfahren zur Tabellenerstellung wie folgt charakterisiert werden:[1)]

1. Bedingungen werden ermittelt und aufgelistet.

2. Aktionen werden ermittelt und aufgelistet.

3. Alle elementaren Bedingungsanzeigerkombinationen werden erstellt und lexikographisch geordnet, so dass die obere Hälfte einer Entscheidungstabelle in kanonischer Normalform entsteht.

4. Regel für Regel werden die Aktionsanzeiger eingetragen.

5. Die entstandenen einfachen Regeln werden so weit wie zulässig in komplexe Regeln transformiert.

6. Es wird auf Redundanz und Vollständigkeit geprüft.

Auch wenn in der Entscheidungstabellen-Literatur nicht darauf hingewiesen wird, ist evident, dass das klassische Verfahren die Theorie der kanonischen alternativen Normalformen aus der mathematischen Logik benutzt, um die Vollständigkeit der Entscheidungstabelle zu einem sehr frühen Zeitpunkt in der Tabellenerstellung sicherzustellen. Die Menge aller elementaren Bedingungsanzeigerkombinationen ist eine Analogie zu den Elementarkonjunktionen einer Aussagenfunktion in kanonisch alternativer Normalform.

Die in der Literatur zu den Beschreibungen dieses Verfahrens angegebenen Beispiele beinhalten selten Probleme, die mehr als 20 elementare Bedingungsanzeigerkombinationen erfordern, so dass die als Zwischenstufe der Tabellenerstellung entstehende Normalform-Tabelle noch sehr übersichtlich ist.

Den Anforderungen der Praxis hält das klassische Verfahren jedoch kaum stand. Nicht selten werden in der Praxis komplexe Probleme bearbeitet, an denen das klassische Verfahren schon deshalb scheitert, weil mehrere hundert oder auch mehr als tausend elementare Bedingungsanzeigerkombinationen berücksichtigt werden müßten,[2)] Der damit und mit den darauf folgenden Teilschritten der Tabellenerstellung verbundene Aufwand ist nicht mehr vertretbar.

---

1) Vgl. London (Decision tables), S. 43 ff.

2) So weist beispielsweise eine Entscheidungstabelle mit 6 Bedingungen mit jeweils 3 Anzeigern bereits 729 elementare Bedingungsanzeigerkombinationen auf.

Ein entscheidender Mangel des klassischen Verfahrens ist vor allem in dessen Überbetonung der Entscheidungstabellen-Syntax zu sehen. Bedingungsanzeigerkombinationen, obwohl bei logischer Abhängigkeit der Bedingungen oft ohne semantische Relevanz, werden überflüssigerweise in das Entscheidungstabellenformular eingetragen; bei bestimmten Bedingungskonstellationen erkennbar irrelevante Bedingungen werden dennoch mit allen Bedingungszuständen in jeder syntaktisch zulässigen Bedingungsanzeigerkombination dokumentiert. Das klassische Verfahren unterstützt im Übrigen auch kaum den Lernprozess, der sich bei der Tabellenerstellung unter unvollkommener Information über das Entscheidungsfeld eigentlich vollziehen müsste. Der Ersteller einer Entscheidungstabelle, der nach säuberlicher Formulierung von fünfzig elementaren Bedingungsanzeigerkombinationen erkennt, dass er noch eine Bedingung übersehen hat, wird angesichts der Wertlosigkeit seiner vorherigen Bemühungen kaum zu noch intensiverer Problemanalyse angeregt, da er sich durch jede neue Erkenntnis regelrecht bestraft fühlen muss.

Nur ein Tabellenerstellungsverfahren, das der semantischen Dimension der Entscheidungstabelle Vorrang vor der syntaktischen einräumt, kann den Anforderungen der Praxis gerecht werden.

1.2.4.2 Tabellenerstellung durch fortschreitende Regelentwicklung [1)]

Offensichtlich aus der Praxis[2)] für die Bedürfnisse der Praxis wurde ein anderes Verfahren zur Tabellenerstellung entwickelt, das in höchstem Maße sicherstellt, dass syntaktischen Erfordernissen bei der Tabellenerstellung nur im Rahmen ihrer semantischen Relevanz Rechnung getragen wird. Das Verfahren der "fortschreitenden Regelentwicklung" soll hier als Kernstück einer optimalen Strategie zur Tabellenerstellung unter unvollkommener Information über das abzubildende Entscheidungsfeld verwendet werden.

Diese Strategie verfolgt das Ziel, erkennbare syntaktische Gesetzmäßigkeiten bei der Tabellenerstellung zur Informationsgewinnung über noch unbekannte Bestandteile des Entscheidungsfeldes auszunutzen, und ver-

---

1) Die ausführlichste Beschreibung dieses Verfahrens gibt London (Decision tables), S. 67 ff In Ansätzen ist das Verfahren bereits dargestellt in NCC (Decision tables), S. 23

2) Die dem Verfasser vorliegende Quelle ältesten Datums sind Unterlagen zu einem internen Entscheidungstabellen-Seminar bei Bell of Canada, die dem Verfasser 1970 übergeben wurden. Vgl. Bell of Canada (Decision tables). Vgl. ferner Sterbakov (Last "Y" method), S. 1 f.

sucht, dieses Ziel durch einen ständigen Wechsel zwischen der Bildung syntaktischer Einheiten und ihrer Verifikation zu erreichen. Kennzeichen dieser Strategie ist auch, dass nicht versucht wird, das gesamte Entscheidungsfeld auf Anhieb zu erfassen und abzubilden, sondern dass Kategorien ähnlicher Merkmale gebildet und nacheinander analysiert und abgebildet werden. Zunächst werden Bedingungen erfasst und abgebildet, danach Aktionen und schließlich Entscheidungsregeln. Es ist zudem sichergestellt, dass Erkenntnisse, die während der Abbildung einer Merkmalskategorie über andere Merkmalskategorien gewonnen werden, auch noch berücksichtigt werden können.

Der Vorgang der Erstellung einer Entscheidungstabelle nach der beschriebenen Strategie kann nur als ein iterativer Prozess verstanden werden, dessen einzelne Phasen in einem Ablaufschaubild nur unvollkommen zu verdeutlichen sind:

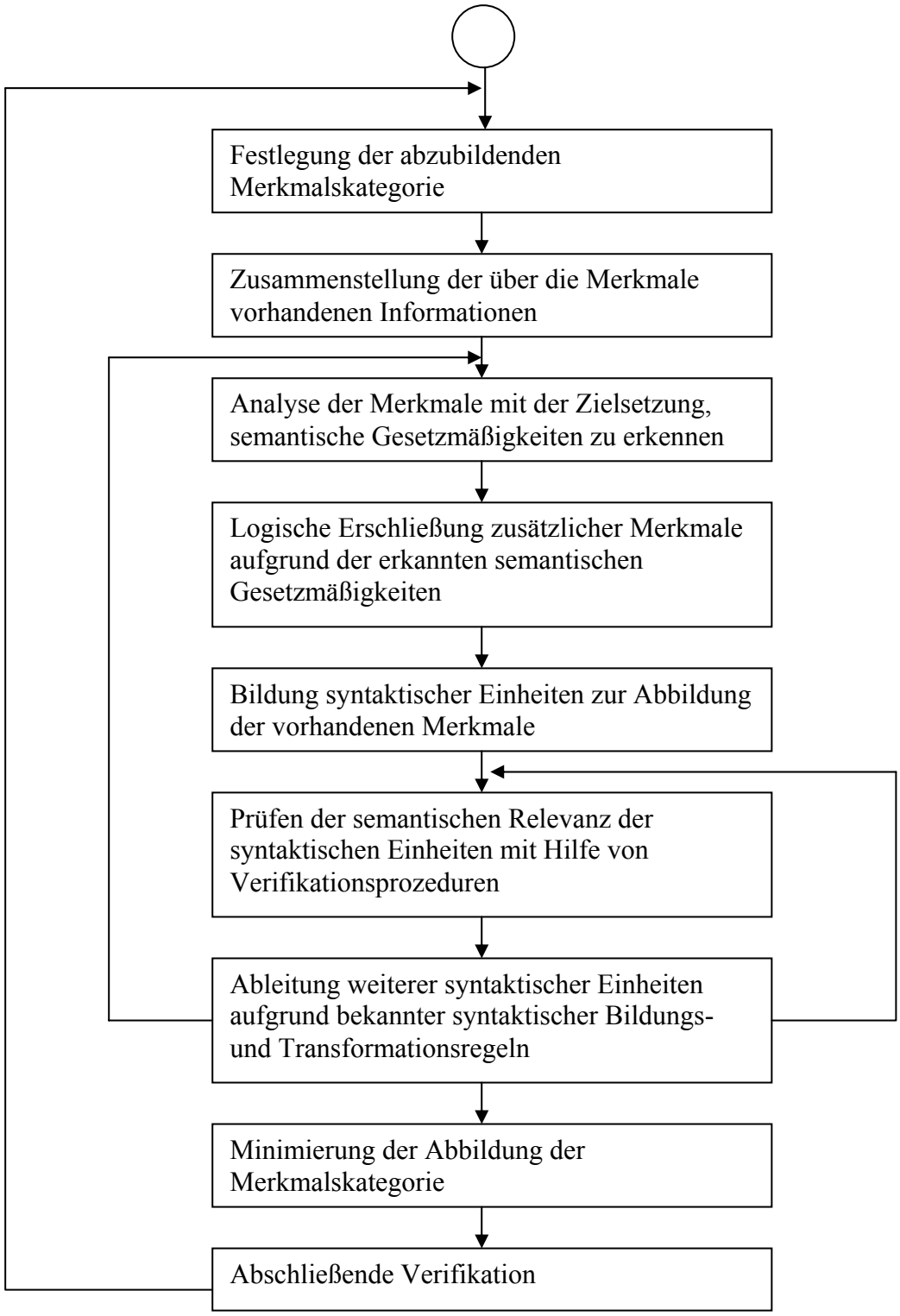

Festlegung der abzubildenden Merkmalskategorie
Zusammenstellung der über die Merkmale vorhandenen Informationen
Analyse der Merkmale mit der Zielsetzung, semantische Gesetzmäßigkeiten zu erkennen
Logische Erschließung zusätzlicher Merkmale aufgrund der erkannten semantischen Gesetzmäßigkeiten
Bildung syntaktischer Einheiten zur Abbildung der vorhandenen Merkmale
Prüfen der semantischen Relevanz der syntaktischen Einheiten mit Hilfe von Verifikationsprozeduren
Ableitung weiterer syntaktischer Einheiten aufgrund bekannter syntaktischer Bildungs- und Transformationsregeln
Minimierung der Abbildung der Merkmalskategorie
Abschließende Verifikation

Abb. 4: Phasen der Erstellung einer Entscheidungstabelle

Ein umfassendes Beispiel, das der Demonstration aller Vorteile des beschriebenen Verfahrens dienen konnte, würde den Rahmen dieser Arbeit sprengen. Deshalb sollen nur die wesentlichen Teilschritte exemplarisch veranschaulicht werden.

Abbildung der Bedingungen und Aktionen

Grundlage der Entscheidungstabellenerstellung sind oft schriftliche Unterlagen oder mündliche Aussagen zum Ablauf eines Informationsverarbeitungsprozesses, aus denen nach einer entsprechenden Analyse Bedingungen erkennbar werden. Diese Bedingungen und ihre mit der Auslösung bestimmter Aktionen verbundenen Bedingungszustände werden dokumentiert.

Danach werden die Bedingungszustände jeder Bedingung daraufhin analysiert, ob ihre Ausprägung aufgrund bestimmter Gesetzmäßigkeiten erfolgte und bei Anwendung dieser Gesetzmäßigkeiten weitere Bedingungszustände logisch erschlossen werden können. Wird beispielsweise festgestellt, dass bestimmte Aktionen in Abhängigkeit vom Ergebnis des Vergleichs zweier Datenelemente A und B ausgeführt werden sollen und sind nur Aktionen für A = B und A > B vermerkt, dann ist der zusätzliche Bedingungszustand A < B leicht logisch zu erschließen. Der zusätzliche Bedingungszustand wird nun auf seine semantische Relevanz innerhalb des abzubildenden Entscheidungsfeldes überprüft. Wenn er plausibel erscheint, wird er zusammen mit den anderen syntaktisch erfasst. Die durch den zusätzlichen Bedingungszustand möglicherweise induzierten zusätzlichen Aktionen werden ebenfalls erfasst.

Die syntaktische Formulierung der Bedingungsanzeiger sollte in der Weise erfolgen, dass spätere Tabellenerstellungs- und Verifikationsprozeduren möglichst auf Syntaxbasis erfolgen können. Eine der wichtigsten Forderungen in diesem Zusammenhang ist, die Bedingungsanzeiger logisch disjunkt zu formulieren.

Wenn in einer Beschreibung zur Ermittlung einer Jahresprämie beispielsweise folgende Formulierungen zu finden sind:

> Beträgt die Betriebszugehörigkeit drei oder mehr Jahre ...
>
> Bei einer Betriebszugehörigkeit von mindestens einem Jahr ...
>
> Gehört der Arbeitnehmer dem Betrieb noch kein volles Jahr an ...

dann würden mit der elementaren syntaktischen Einheit Betriebszugehörigkeit: <1, >=1, >=3

zwar die Bedingungszustände korrekt abgebildet, durch die Überlappung der beiden letzten Anzeiger würden jedoch Tabellenerstellung und -verifikation unnötig erschwert. Die folgende Formulierung bildet ebenso korrekt ab, die Anzeiger sind jedoch logisch disjunkt:

Betriebszugehörigkeit: <1, 1-2, >=3

Logisch disjunkte Anzeiger sind nicht notwendigerweise auch vollständig. Die zuletzt wiedergegebene Formulierung setzt voraus, dass ganzzahlige Daten über die Betriebszugehörigkeit verfügbar sind.

Nach der syntaktischen Erfassung der vorhandenen Bedingungen und Aktionen wird versucht, zusätzliche entscheidungsrelevante Bedingungen und Aktionen zu ermitteln. Eine dazu geeignete Methode, durch eine Analyse der logischen Beeinflussungen der Aktionen durch die Bedingungen und vice versa zusätzliche Bedingungen und Aktionen herauszufinden, wurde bereits beschrieben.[1)]

Festlegung der Basisfolge von Bedingungen und Aktionen

Bei *logischer Unabhängigkeit* der Bedingungen kann jede beliebige Annordnung der Bedingungen als Basisfolge für die Entscheidungstabelle gewählt werden. Bei *logischer Abhängigkeit* der Bedingungen muss geprüft werden, ob die Art der logischen Abhängigkeit eine bestimmte Anordnung der Bedingungen erforderlich macht. Eine bestimmte Klasse logisch abhängiger Bedingungen beispielsweise resultiert aus der Notwendigkeit, zeitliche Beziehungen bei der Bedingungsprüfung zu beachten. Soll beispielsweise im Rahmen einer rechnergestützten Auftragsbearbeitung mit Hilfe einer mit Prüfziffer versehenen Kundennummer der Stammsatz des Kunden aus der Stammdatei ermittelt werden, dann könnten die erforderlichen Bedingungen wie folgt formuliert werden:

Bl Stammdaten vorhanden?: J, N
B2 Kundennummer gültig? : J, N

Leicht ist ersichtlich, dass die wiedergegebene Bedingungsfolge nicht die Basisfolge einer Entscheidungstabelle bilden kann, da die Prüfung auf Vorhandensein der Stammdaten die vorherige Gültigkeitsprüfung der Kundennummer erforderlich macht. Bei umgekehrter Reihenfolge könnte diese zeitliche Beziehung bei der Bedingungsprüfung berücksichtigt werden.

---

1) Vgl. Abschnitt 1.2.2.2.1

Die Basisfolge der Aktionen muss so festgelegt werden, dass alle plausibel erscheinenden Aktionsfolgen als Teilmenge ohne die Notwendigkeit einer Umordnung gebildet werden können. In vielen Fällen lassen sich auch bei den Aktionen logische Abhängigkeiten zeitlicher Art feststellen, die eine Aktionsausführung in bestimmter Reihenfolge erforderlich machen und dadurch die Anordnung der Aktionen innerhalb der Basisfolge bestimmen.

Die Abbildung der Entscheidungsregeln nach der Methode der "fortschreitenden Regelentwicklung"[1)] beginnt auf der Basis der in den vorhergehenden Phasen ermittelten und geordneten Bedingungen und Aktionen, lässt jedoch durchaus noch Spielraum für eine Berücksichtigung zusätzlicher Bedingungen und Aktionen, die erst während der Regelentwicklung entdeckt werden. Die Methode soll an einem formalen Beispiel illustriert werden.

Es sollen folgende Bedingungen und Aktionen als elementare syntaktische Einheiten der Entscheidungstabelle bereits definiert worden sein:

Bedingung a mit [a] = {al,a2,a3,a4}
Bedingung b mit [b] = {bl,b2,b3}
Bedingung c mit [c] = {cl,c2,c3,c4,c5}
Bedingung d mit [d] = {dl,d2}

Aktion α mit [α] = {X}
Aktion β mit [β] = {X}
Aktion γ mit [γ] = {X}

Die Bedingungsanzeiger jeder Bedingung sind logisch disjunkt und vollständig. Alle Bedingungen und Aktionen sind bereits geordnet. Es wird dann zunächst aus dem jeweils ersten Anzeiger jeder Bedingung eine Bedingungsanzeigerkombination gebildet:

$$\left\{ \begin{array}{l} a1 \\ b1 \\ c1 \\ d1 \end{array} \right\}$$

1) Vgl. London (Decision tables), S. 67 ff.

Diese syntaktische Bedingungsanzeigerkombination ist nun auf ihre semantische Relevanz zu prüfen, d.h. es ist festzustellen, ob eine der Abbildung entsprechende Bedingungskonstellation in der Realität überhaupt auftreten kann. Angenommen, die Bedingungen a und d sind derart voneinander logisch abhängig, dass in allen Fällen, in denen Bedingung a den Zustand al hat, bei Bedingung d nur noch der Zustand d2 möglich ist. Dann bildet die formulierte Bedingungsanzeigerkombination eine logisch unzulässige (semantisch irrelevante) Bedingungskonstellation ab und braucht in der Entscheidungstabelle nicht berücksichtigt zu werden. Gleichzeitig können aber alle weiteren formalen Bedingungsanzeigerkombinationen aus dem Prozess der Regelerstellung eliminiert werden, durch die die Bedingungskonstellation abgebildet wird.

$$\left\{\begin{array}{c} a1 \\ - \\ - \\ d1 \end{array}\right\}$$

Es sind dies:

$$\left\{\begin{array}{cccccccccccccc} a1 & a1 & a1 & a1 & a1 & a1 & a1 & a1 & a1 & a1 & a1 & a1 & a1 & a1 \\ b1 & b1 & b1 & b1 & b2 & b2 & b2 & b2 & b2 & b3 & b3 & b3 & b3 & b3 \\ c2 & c3 & c4 & c5 & c1 & c2 & c3 & c4 & c5 & c1 & c2 & c3 & c4 & c5 \\ d1 & d1 & d1 & d1 & d1 & d1 & d1 & d1 & d1 & d1 & d1 & d1 & d1 & d1 \end{array}\right\}$$

Als nächstes wird die folgende Bedingungsanzeigerkombination gebildet:

$$\left\{\begin{array}{c} a1 \\ b1 \\ c1 \\ d2 \end{array}\right\}$$

Sie wird dadurch aus der ersten Bedingungsanzeigerkombination abgeleitet, dass der untersten Bedingung der zweite Bedingungsanzeiger zugeordnet wird.

Die semantische Relevanzprüfung soll ergeben, dass diese Bedingungsanzeigerkombination eine logisch zulässige Bedingungskonstellation abbildet. Die Aktionen β und γ sollen bei Eintreffen dieser Konstellation ausgeführt werden. Es kann somit folgende Entscheidungsregel formuliert werden:

		R1
B1	Bedingung a	a1
B2	Bedingung b	b1
B3	Bedingung c	c1
B4	Bedingung d	d2
A1	Aktion α	
A2	Aktion β	X
A3	Aktion γ	X

Es schließt sich nun eine weitere semantische Analyse an. Ausgehend von der definierten Aktionsfolge wird geprüft, ob alle als relevant ausgewiesenen Bedingungen auf die Auslösung der definierten Aktionsfolge einen Einfluss ausüben. Dazu kann gedanklich bei jeder Bedingung der vorhandene Anzeiger nacheinander durch alle anderen Anzeiger ersetzt werden. Müsste die Aktionsfolge auch bei nur einer der so entstehenden Bedingungsanzeigerkombinationen geändert werden, dann ist nachgewiesen, dass die Bedingung, bei der eine Anzeigervariation durchgeführt wurde, Einfluss ausübt. Bleibt die Aktionsfolge dagegen gültig, dann ist nachgewiesen, dass die Bedingung, bei der die Anzeigervariation durchgeführt wurde, keinen Einfluss ausübt. Sie kann im Rahmen der untersuchten Regel durch "-" als indifferent gekennzeichnet werden. Für das Beispiel soll angenommen werden, dass sich Bedingung c als indifferent erweist. Als erste Regel der zu erstellenden Entscheidungstabelle kann dann endgültig formuliert werden:

		1
B1	Bedingung a	a1
B2	Bedingung b	b1
B3	Bedingung c	-
B4	Bedingung d	d2
A1	Aktion α	
A2	Aktion β	X
A3	Aktion γ	X

Von dieser Regel abgeleitet, muss als nächste Bedingungsanzeigerkombination untersucht werden:

$$\left\{\begin{matrix} a1 \\ b2 \\ c1 \\ d2 \end{matrix}\right\}$$

Das resultiert daraus, dass d2 der letzte Anzeiger der Bedingung d und Bedingung c indifferent ist. Die nächste Anzeigervariation muss also bei Bedingung b erfolgen. Die sich dann syntaktisch ergebende Anzeigerkombination

$$\left\{\begin{matrix} a1 \\ b2 \\ c1 \\ d1 \end{matrix}\right\}$$

war bereits als logisch irrelevant erkannt worden, so dass erst wieder die Anzeigerkombination

$$\left\{\begin{matrix} a1 \\ b2 \\ c1 \\ d2 \end{matrix}\right\}$$

auf ihre semantische Relevanz überprüft werden muss. Es soll angenommen werden, dass bei dieser Prüfung eine weitere, bisher noch nicht bekannte Aktion mit der syntaktischen Formulierung

$$\text{Aktion } \alpha \text{ mit } [\delta] = \{x\}$$

als für die abgebildete Bedingungskonstellation erforderlich erkannt wird. Zusätzlich soll Aktion α ausgeführt werden, und zwar vor Aktion δ

Wird zusätzlich noch unterstellt, dass sich keine Bedingung als indifferent erweist, dann kann die zweite Regel wie folgt ergänzt werden:

		1	2
B1	Bedingung a	a1	a2
B2	Bedingung b	b1	b2
B3	Bedingung c	-	c1
B4	Bedingung d	d2	d2
A1	Aktion α		X
A2	Aktion β	X	
A3	Aktion γ	X	
A4	Aktion δ		X

Die nächsten fünf Regeln sollen sich wie folgt ergeben:

		1	2	3	4	5	6	7
B1	Bedingung a	a1	a2	a3	a4	a5	a6	a7
B2	Bedingung b	b1	b2	b2	b2	b2	b2	b3
B3	Bedingung c	-	c1	c2	c3	c4	c5	-
B4	Bedingung d	d2	d2	d2	d2	d2	d2	d2
A1	Aktion α		X			X	X	
A2	Aktion β	X		X				
A3	Aktion γ	X			X			X
A4	Aktion δ		X					

Da mit Regel 7 auch bei Bedingung b alle Anzeigerformen durchgeprüft sind, ist als nächste Bedingungsanzeigerkombination zu untersuchen.

$$\left\{\begin{matrix} a2 \\ b1 \\ c1 \\ d1 \end{matrix}\right\}$$

Angenommen, es handelt sich dabei zwar um die Abbildung einer logisch zulässigen Bedingungskonstellation: Eine eindeutige Zuordnung einer Aktionsfolge ist jedoch trotzdem nicht möglich, da die Auswirkung einer zusätzlichen Bedingung bisher übersehen wurde. Diese zusätzliche Bedingung wird syntaktisch in folgender Weise definiert:

Bedingung e mit [e] = {el, e2}.

Erst zu den Anzeigerkombinationen

$$\left\{\begin{matrix} a2 \\ b1 \\ c1 \\ d1 \\ e1 \end{matrix}\right\}$$

ist die Zuordnung der Aktionsfolge möglich.

Aktion α
Aktion β

Nachdem sichergestellt ist, dass sich die neue Bedingung in Bezug auf die bereits beschriebenen Bedingungskonstellationen indifferent verhält und außerdem die Bedingungen b, c und d als indifferent erkannt

wurden, kann die neue Regel 8 ergänzt werden.

		1	2	3	4	5	6	7	8
B1	Bedingung a	a1	a2	a3	a4	a5	a6	a7	a2
B2	Bedingung b	b1	b2	b2	b2	b2	b2	b3	-
B3	Bedingung c	-	c1	c2	c3	c4	c5	-	-
B4	Bedingung d	d2	d2	d2	d2	d2	d2	d2	-
B5	Bedingung e	-	-	-	-	-	-	-	e1
A1	Aktion α		X			X	X		
A2	Aktion β	X		X					X
A3	Aktion γ	X			X			X	X
A4	Aktion δ		X						

Alle weiteren Regeln werden nach der beschriebenen Methode formuliert. Es soll angenommen werden, dass als Endergebnis eine Entscheidungstabelle mit den folgenden 15 Regeln entsteht:

		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
B1	Bedingung a	a1	a2	a3	a4	a5	a6	a7	a2	a2	a3	a3	a3	a3	a3	a4
B2	Bedingung b	b1	b2	b2	b2	b2	b2	b3	-	-	-	-	-	-	-	-
B3	Bedingung c	-	c1	c2	c3	c4	c5	-	-	-	c1	c2	c3	c4	c5	-
B4	Bedingung d	d2	d2	d2	d2	d2	d2	d2	-	-	-	-	-	-	-	-
B5	Bedingung e	-	-	-	-	-	-	-	e1	e2	-	-	-	-	-	-
A1	Aktion α		X			X	X				X	X	X	X		X
A2	Aktion β	X		X					X	X	X	X	X	X	X	X
A3	Aktion γ	X			X			X	X	X					X	X
A4	Aktion δ		X							X						X

Eine abschließende Verifikation, die sich bei den für das Beispiel geltenden Prämissen auf eine syntaktische Redundanz- und Vollständigkeitsprüfung im Quadranten der Bedingungsanzeigerkombinationen beschränken kann, ergibt Eindeutigkeit und 210 abgedeckte Bedingungskonstellationen. Da formal 240 elementare Bedingungskonstellationen existieren, muss der Verbleib der nicht abgebildeten 30 Bedingungskonstellationen geklärt werden.

Durch die logische Abhängigkeit zwischen den Bedingungen a und d und durch die neu hinzugekommene Bedingung e sind jedoch 3 • 5 • 2 = 30 Konstellationen unzulässig, so dass die Entscheidungstabelle damit als semantisch vollständig gelten kann.

1.2.4.3 Erstellung des Entscheidungstabellenverbundes

Die Erstellung eines Entscheidungstabellenverbundes erschöpft sich nicht darin, vorhandene Einzeltabellen durch Ergänzen spezieller Verknüpfungsaktionen untereinander zu verbinden. Vielmehr besteht das Problem darin, ein umfangreiches Entscheidungsfeld in Gruppen logisch zusammengehörender Bedingungen und Aktionen zu zergliedern und diese Gruppen prozedural so anzuordnen, dass den semantischen Erfordernissen des Entscheidungsprozesses Rechnung getragen wird. Jede Bedingungs-/Aktionsgruppe kann dann zusammen mit den zugehörigen Regeln als eine Teil-Entscheidungstabelle des Verbundes dokumentiert werden.

Bei der Erstellung eines Entscheidungstabellenverbundes können zueinander konträre Zielsetzungen verfolgt werden. So wird ein Entscheidungstabellenverbund, der unter der Zielsetzung der Minimierung seiner syntaktischen Einheiten entwickelt wurde, eine andere Struktur aufweisen als ein Verbund, der die Minimierung der dokumentierten Entscheidungsprozesse anstrebt. Die jeweilige Zielsetzung wird in der Regel unter Beachtung bestimmter Nebenbedingungen angestrebt, die Tabellengröße und Tabelleninhalt betreffen. Tabellen mit mehr als ca. 30 Regeln gelten beispielsweise als zu groß aus pragmatischer Sicht.[1)] Andererseits wird auch nicht eine minimale Tabellengröße angestrebt, da logisch abgrenzbare Phasen des Entscheidungsprozesses aus Gründen der Übersichtlichkeit gerne in Form einer einzigen Entscheidungstabelle dokumentiert werden.

Nur wenige formale Verfahren, die bei der Erstellung eines Entscheidungstabellenverbundes angewendet werden können, sind bisher bekannt.
So eignet sich beispielsweise das bereits dargestellte[2)] Zergliederungsverfahren durch kombinierte Bedingungs-/Aktionsaufteilung auch zur Erstellung eines Entscheidungstabellenverbundes. Folgende Teilschritte sind erforderlich:[3)]

1. Bedingungen und Aktionen des Entscheidungsfeldes werden matrixförmig angeordnet. Dabei wird die Reihenfolge von Bedingungen und Aktionen in Abhängigkeit von den gegebenen zeitlichen und logischen Abhängigkeiten gewählt.

---

1) Nach der Erfahrung des US Bureau of the Census liegt bei begrenzten Entscheidungstabellen die obere Grenze bei 24 - 32 Regeln. Vgl. O'Brien (Computerizing administrative operations), S. 68
2) Vgl. Abschnitt 1.2.2.2.1
3) Vgl. Strunz (Parsing), S. 366 ff.

2. Der Einfluss jeder Bedingung auf jede Aktion wird durch "X" als Matrixelement angezeigt.

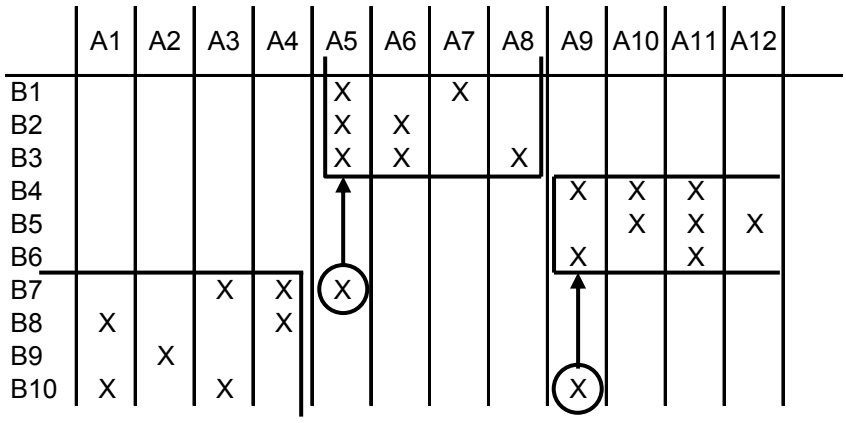
3. Bedingungen und Aktionen werden so gruppiert, dass die logischen Beziehungen der Bedingungen und Aktionen innerhalb einer Gruppe maximiert und die logischen Beziehungen der Bedingungen einer Gruppe zu den Aktionen anderer Gruppen minimiert werden.

4. Zu den ermittelten Bedingungs-/Aktionsgruppen werden entsprechende Entscheidungstabellen erstellt.

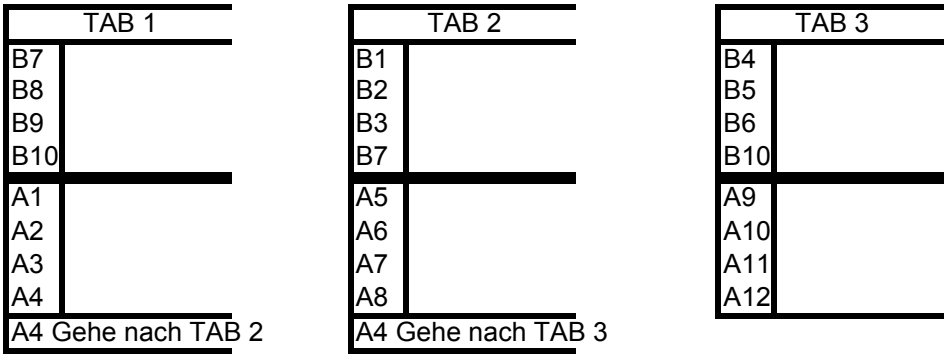
5. Die entstandenen Tabellen werden prozedural so untereinander verknüpft, dass die erforderliche Reihenfolge der Aktionsausführung gewährleistet ist.

	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12
B1					X		X					
B2					X	X						
B3					X	X		X				
B4									X	X	X	
B5										X	X	X
B6									X		X	
B7			X	X	X							
B8	X			X								
B9		X										
B10	X		X						X			

Diese Matrix lässt deutliche Ansatzpunkte für eine sinnvolle Gestaltung eines Entscheidungstabellenverbundes erkennen, da sich aufgrund der eingetragenen logischen Beziehungen drei Bedingungs-/Aktionsgruppen abzeichnen:



Etwas störend wirken sich die logischen Beziehungen zwischen Bedingung 7/Aktion 5 sowie Bedingung 10/Aktion 9 aus. Sie bewirken, dass die Bedingungen 7 und 10 jeweils in zwei Entscheidungstabellen des Verbundes aufgeführt werden müssen. Den drei Entscheidungstabellen des Verbundes werden dann die Bedingungen und Aktionen wie folgt zugeordnet:



Chapin[1)] beschreibt ein anderes Verfahren zur Bestimmung logisch zusammengehörender Bedingungs- und Aktionsgruppen, das sich an nur semantisch erkennbaren Merkmalen des Problems orientiert. Er empfiehlt, die erkennbaren Datenstrukturen hierarchisch zu gliedern und die Bedingungen und Aktionen so zu gruppieren, dass nicht Daten unterschiedlicher hierarchischer Stufe in einer Entscheidungstabelle verarbeitet werden. Ergänzend wird empfohlen, der Gruppierung eine horizontale Gliederung der Verarbeitungsvorgänge von Daten jeder hierarchischen Stufe in Initialisierungsprozeduren, Verarbeitungsprozeduren im engeren Sinn und abschließende Aufbereitungsroutinen zugrunde zu legen.

Speziell für Realzeit-Anwendungen empfiehlt Chapin die zusätzliche Berücksichtigung von Prioritäten, die bei der Verarbeitung der Daten beachtet werden müssen oder die logische Verknüpfung verschiedener Datenarten durch ihren spezifischen Inhalt einer Gruppierung zugrunde zu legen.

Auch die bereits erläuterten Verfahren der Kombination von Entscheidungstabellen[2)] können bei der Erstellung eines Entscheidungstabellenverbundes nützlich sein. Teiltabellen des Verbundes lassen sich damit wieder zusammenfassen, wenn die Voraussetzungen gegeben sind und es zur Erfüllung des Gestaltungszieles sinnvoll erscheint.

---

1) Chapin (Parsing), S. 508 ff.
2) Vgl. dazu Abschnitt 1.1.2.2.2

1.3 Pragmatik der Entscheidungstabelle

Die pragmatische Dimension einer Sprache betrifft ihr Verhältnis zu den Anwendern der Sprache. [1] Der Anwendung von Entscheidungstabellen als Hilfsmittel der Gestaltung rechnergestützter Informationssysteme ist der gesamte zweite Teil dieser Arbeit gewidmet, so dass an dieser Stelle nur einige grundsätzliche Bemerkungen gemacht werden sollen.

Entscheidungstabellen beschreiben strukturelle Aspekte von Entscheidungsprozessen, so dass ihre formale Charakterisierung als "Strukturspräche" durchaus angemessen ist. Allein durch die Beschreibung ihrer Struktur werden Entscheidungsprozesse jedoch nicht übertragbar, es wird eine zusätzliche Sprache zur Beschreibung der spezifischen Bedingungen mit ihren Zuständen und der Aktionen mit ihren Ausprägungen benötigt. Dieser Sprache die Stellung einer *Primärsprache* einzuräumen und die Gesamtheit der syntaktischen, semantischen und pragmatischen Regeln in der Entscheidungstabellentechnik als *Sekundärsprache* zu bezeichnen, ist vertretbar, wenn berücksichtigt wird, dass zwar Entscheidungsprozesse auch ohne Entscheidungstabellen beschrieben werden können, umgekehrt die Formalisierung realer Entscheidungsprozesse unter ausschließlicher Verwendung von Entscheidungstabellen nicht möglich ist.

Eine vollständige Darstellung der syntaktischen, semantischen und *pragmatischen Dimension* der Entscheidungstabellensprache wäre möglich, ohne auf eine Primärsprache Bezug zu nehmen. Bedingungen und Aktionen werden dann ausschließlich durch formale Substitute der Form

> Bedingung a mit [a] = {a1, a2, ... ,an}
> Aktion ω mit [ω] ={ωl,ω2, ...ωm}

repräsentiert. Wenn in den bisherigen Kapiteln in Einzelfällen von der Verwendung solcher formaler Substitute abgewichen wurde und Ausdrücke einer Primärsprache einbezogen wurden, dann geschah dies nur im Interesse einer besseren Veranschaulichung.

Erst recht gilt für die spätere Erörterung der pragmatischen Dimension im Rahmen der Gestaltung rechnergestützter Informationssysteme, dass eine isolierte Betrachtung der Entscheidungstabellensprache ohne Bezugnahme auf die mit ihr kombinierbaren Primärsprachen nur unbefriedigende Ergebnisse bringen würde. Gerade aus dem Zusammenwirken der Entscheidungstabellensprache mit einer Primärsprache ergeben sich interessante pragmatische Auswirkungen, die - weil höchst praxisrelevant - unbedingt analysiert werden müssen.

---

1) Vgl. Morris (Theory of signs), S. 30

2. Ansatz zu einer Theorie der Mehrtreffer-Entscheidungstabelle

Der erste Hinweis auf die "Mehrtreffer-Entscheidungstabelle" [1] geht auf Barnard[2] zurück, der bei der Gestaltung eines Standard-Softwarepakets zur Auswertung von Dateien auf die Entscheidungstabelle zur Formulierung der Auswahllogik nicht verzichten wollte, jedoch nach Möglichkeiten suchte, den Aufwand zur maschinellen Verarbeitung einer Entscheidungstabelle zu verringern. Er entdeckte dabei, dass die Anzahl der Regeln einer Entscheidungstabelle verringert werden kann, wenn zugelassen wird, dass eine Bedingungskonstellation und die ihr zugeordnete Aktionsfolge in mehr als einer Regel abgebildet werden. Die so entstehende Entscheidungstabelle enthält - gemessen an den Bildungsregeln einer Eintreffertabelle - Mehrdeutigkeiten. Durch klar definierte Interpretationsregeln werden ungünstige Auswirkungen solcher "Mehrdeutigkeiten" jedoch vermieden.

Der Verfasser empfahl 1970[3] - zum Zwecke der einfacheren Zergliederung großer Entscheidungstabellen - Bedingungen und Aktionen matrixförmig anzuordnen und aktionsbezogene Bedingungsanzeigerkombinationen einzutragen. Rückschauend betrachtet, wurde damit ebenfalls die Mehrtreffer -Entscheidungstabelle vorgeschlagen, wenn auch mit anderer Zielsetzung und in einem vom Standardformat der Entscheidungstabelle abweichenden Format.

In den Jahren 1970 - 1972 entstanden in Europa unabhängig voneinander drei Entscheidungstabellenvorübersetzer, die auch Mehrtreffer – Entscheidungstabellen zu verarbeiten in der Lage waren. Soweit rekonstruierbar, waren die Motive der Softwareentwickler zur Einbeziehung der Mehrtreffer -Entscheidungstabelle in das Verarbeitungsspektrum ihrer Vorübersetzer durchaus unterschiedlich.

---

1) Der Begriff "Mehrtreffer–Entscheidungstabelle" (vgl. auch FNI (Entscheidungstabelle), S. 7) wurde in Anlehnung an die von der CODASYL Decision Tables Task Group nach einem Vorschlag von Muntz verwendeten Bezeichnung "multiple hit decision table" gewählt. Vgl. Muntz (Multiple choice decisions tables), S.3
   Die Bezeichnung "Mehrtreffer–Entscheidungstabelle" soll darauf hinweisen, dass eine Bedingungskonstellation in der Realität durch mehrere Regeln der Entscheidungstabelle gemeinsam abgebildet werden kann.
   Leider ist in deutschen Veröffentlichungen und Firmenbroschüren auch der Begriff "mehrdeutige Entscheidungstabelle" zu finden. Dieser Ausdruck ist irreführend, da die Mehrtreffer-Entscheidungstabelle ebenso eindeutig interpretierbar ist wie die Eintreffer–Entscheidungstabelle.
   Auf die Verwendung dieses Ausdrucks wird daher verzichtet
2) Barnard (New rule mask technique), S. 153 f .. Die Veröffentlichung erfolgte 1969; in dem erwähnten Softwareprodukt war die Mehrtreffer–Entscheidungstabelle bereits 1967 implementiert. Vgl. Bernard (NITA)
3) Vgl. Strunz (Zergliederung), S. 5, vgl. ferner Strunz (Eine Methode zur Zergliederung) und Strunz (Parsing)

Die günstigere Beschreibung der Logik komplexer Plausibilitätsprüfungen war offensichtlich das Hauptmotiv zur Einbeziehung von Mehrtreffer -Entscheidungstabellen in DETAB/GT[1)], der in der Schweiz entwickelt wurde.

Günstigere Verarbeitungsmöglichkeiten der Mehrtreffer-Entscheidungstabelle im Vergleich zur Eintreffer-Entscheidungstabelle sowie eine einfachere Tabellenerstellung können als Motive für die Erweiterung von AGENTA, einem in der Bundesrepublik Deutschland entstandenen Vorübersetzer, um die Fähigkeit zur Verarbeitung von Mehrtreffer-Entscheidungstabellen gelten.[2)]

Die Fähigkeit zur Verarbeitung von Mehrtreffer-Entscheidungstabellen in VORELLE[3)], einem ebenfalls in der Bundesrepublik Deutschland ab 1970 entwickelten Vorübersetzer, entstand aus der Absicht, die vom Verfasser vorgeschlagene Zergliederungstechnik mit Hilfe der Entscheidungsmatrix zu automatisieren.[4)]

Rasch wurde erkannt, dass es entwicklungstechnisch günstiger war, auf die automatische Zergliederung zu verzichten und die Entscheidungsmatrix direkt zu verarbeiten.

In den USA wurde die Mehrtreffer-Entscheidungstabelle im Rahmen einer Forschungsarbeit zur Entwicklung einer Programmiersprache "entdeckt", mit der Prozesse zur automatischen Diagnose der Funktionsfähigkeit beliebiger Maschinen beschrieben werden können.[5)] Als Ergebnis automatischer Diagnosen besteht häufig die Notwendigkeit, in Abhängigkeit vom Ergebnis der Messung vieler Variablen Meldungen auszugeben, die mehr vom Zustand einzelner Variablen als von der *Kombination* ihrer Zustände induziert werden.[6)] Die Verwendung der Eintreffer-Entscheidungstabelle würde die Anzahl erforderlicher Entscheidungsregeln in diesem Fall erheblich größer werden lassen als bei der Mehrtreffertabelle.

---

1) Vgl. Thurner (Entscheidungstabellen), S. 65
2) Vgl. Dathe (Conversion of decision tables), S. 906 ff. und software ag (agenta), S. 3
3) Unter der Projektleitung des Verfassers von Mitarbeitern des mbp, Mathematischer Beratungs- und Programmierungsdienst GmbH, entwickelt
4) Vgl. Strunz (Voruntersuchungen), S. 66
5) Vgl. Loveman (CTL, S. 1 ff.
6) Vgl. Muntz (Multiple choice decision tables), S. 3

In den wenigen Jahren ihrer Existenz hat sich die Mehrtreffer–Entscheidungstabelle aus pragmatischer Sicht für bestimmte Anwendungsgebiete der Eintreffer-Entscheidungstabelle überlegen gezeigt.[1)] Das ist Anlass genug, den Ansatz zu einer Theorie der Mehrtreffer-Entscheidungstabelle vorzulegen, die die syntaktische, semantische und pragmatische Dimension dieser Tabellenart im Vergleich zur Eintreffer-Entscheidungstabelle untersucht und systematisiert.

2.1 Syntax der Mehrtreffer - Entscheidungstabelle

Die Mehrtreffer-Entscheidungstabelle ist in vielen Bereichen der Syntax mit der Eintreffer-Entscheidungstabelle identisch, in anderen wiederum weist sie erhebliche Abweichungen auf. Die Kenntnis der erläuterten Syntax der Eintreffer-Entscheidungstabelle voraussetzend, können sich die folgenden Ausführungen auf eine Beschreibung der Abweichungen beschränken.

2.1.1 Bildungsregeln

Bedingungen und Aktionen gelten auch bei der Mehrtreffertabelle als elementare Einheiten, ebenso lassen sich Bedingungs- und Aktionszeigerkombinationen, Entscheidungsregeln sowie Entscheidungstabellen und der Entscheidungstabellenverbund als höhere syntaktische Einheiten in der bekannten Weise bilden.

Unterschiede lassen sich beim Tabellenformat, bei den Regeln für die Bildung der Entscheidungstabelle und bei der syntaktischen Klassifikation aufzeigen.

2.1.1.1 Format der Mehrtreffer-Entscheidungstabelle

Die Mehrtreffer-Entscheidungstabelle kann, ebenso wie die Eintreffer -Entscheidungstabelle, im bekannten Format mit vertikaler Anordnung der Entscheidungsregeln beschrieben werden. Als Alternative ist jedoch auch ein Darstellungsformat möglich, bei dem die Bedingungen als Zeilenbeschriftungen, die Aktionen als Spaltenüberschriften und die Anzeiger als Elemente einer Matrix eingetragen werden.

So wird es möglich, jeder Aktion die sie auslösenden Bedingungsanzeigerkombinationen exakt zuzuordnen.

---

1) Es sei hier beispielhaft hingewiesen auf Plausibilitätsprüfungen eingegebener Daten, Diagnoseprobleme allgemeiner Art und Auswahlprobleme im Rahmen einer automatischen Arbeitsplanung

Für

Bedingung a mit [a] = {J, N}
Bedingung b mit [b] = (J, N}
Bedingung e mit [e] = {J, N}
Bedingung d mit [d] = { J, N}

und

Aktion α mit [α] = {X)
Aktion β mit [β] ={X)
Aktion γ mit [γ] ={X}
Aktion δ mit [δ] ={X}
Aktion ε mit [ε] =(X)

könnte eine Mehrtreffer -Entscheidungstabelle in Form einer Entscheidungsmatrix folgende Gestalt haben:[1)]

	Aktion α		Aktion β	Aktion γ	Aktion δ	Aktion ε	
Bedingung a	J		N			N	J
Bedingung b	N	J	N				
Bedingung c				J	N		
Bedingung d	J		J	J		J	N

Bei Verwendung komplexer Bedingungen und Aktionen empfiehlt es sich jedoch, bei diesem Format aus Gründen der Lesbarkeit nur die Anzeiger der Bedingungen als Matrixelemente einzutragen und die komplexen Aktionen in einfache Aktionen aufzulösen.

Das gleiche Problem, im Standardformat der Eintreffertabelle dargestellt, ergibt sich wie folgt.

1) Vgl. Strunz (Zergliederung), S. 5

		1	2	3	4	5	6	7
B1	Bedingung a	J	-	N	-	-	N	J
B2	Bedingung b	N	J	N	-	N	-	-
B3	Bedingung c	-	-	-	J	-	-	-
B4	Bedingung d	J	-	J	J	-	J	N
A1	Aktion α	X	X					
A2	Aktion β			X				
A3	Aktion γ				X			
A4	Aktion δ					X		
A5	Aktion ε						X	X

Im Gegensatz zur Matrixdarstellung ist es mit diesem Format zusätzlich möglich, den in der oberen Hälfte abgebildeten Bedingungskonstellationen *Aktionsfolgen* zuzuordnen. Auch können komplexe Aktionen ohne Einschränkung eingetragen werden. Dieses Format soll im Folgenden ebenfalls als "Standardformat" der Mehrtreffer-Entscheidungstabelle gelten.

2.1.1.2 Bildung der Mehrtreffer-Entscheidungstabelle

Es wurde bereits erörtert, dass - im Gegensatz zur Bildung der Eintreffer-Entscheidungstabelle - bei der Mehrtreffertabelle alle syntaktisch zulässigen Entscheidungsregeln erlaubt sind, also auch Entscheidungsregeln mit Bedingungsanzeigerkombinationen, die nicht disjunkt sind.[1)] Selbst Regeln mit identischen Bedingungsanzeigerkombinationen sind erlaubt.

Damit entspricht der formale Aufbau des Bedingungsanzeigerteils dem formalen Aufbau des Aktionsanzeigerteils.

2.1.1.3 Syntaktische Klassifikation von Mehrtreffer-Entscheidungstabellen

Je nach Art der syntaktischen Formulierung von Bedingungen, Aktionen und Anzeigern lassen sich auch bei der Mehrtreffer – Entscheidungstabelle *begrenzte*, *erweiterte* und *gemischte* Entscheidungstabellen unterscheiden. Ebenso ist ihre Darstellung in kanonischer Normalform möglich, dabei verliert sie jedoch ihre Unterscheidungsmerkmale zur Eintreffer-Entscheidungstabelle,

---

1) Vgl. Abschnitt 1.1.1.3.6

da die in einer Normalform-Tabelle aufgeführten elementaren Bedingungsanzeigerkombinationen immer logisch disjunkt sind.

Es lassen sich jedoch vollständige und unvollständige Tabellen unterscheiden.

Def.: Eine Mehrtreffer-Entscheidungstabelle ist *vollständig*, wenn in ihren Regeln jede elementare Bedingungsanzeigerkombination mindestens je einmal repräsentiert und jeder zulässige Aktionsanzeiger Bestandteil mindestens einer der aufgeführten Aktionsanzeigerkombinationen ist.

Beispiel:

Für

Bedingung a mit [a] = {al,a2,a3}
Bedingung b mit [b] = {J,N}
Bedingung c mit [c] = {J,N}

und

Aktion a mit [α] = {X}
Aktion B mit [β] = {βl,β2,β3,β4}

ist die folgende Mehrtreffer-Entscheidungstabelle den erläuterten Syntaxreregeln entsprechend gebildet worden.

		1	2	3	4	5
B1	Bedingung a	a1	-	a2	a3	a3
B2	Bedingung b	-	J	-	-	N
B3	Bedingung c	N	-	N	N	J
A1	Aktion α	X			X	X
A2	Aktion β		β 2	β 1	β 1	β 3

Sie ist jedoch syntaktisch unvollständig, da die elementaren Bedingungsanzeigerkombinationen

{a1, N, J} und {a2, N, J}

nicht repräsentiert sind und keine der aufgeführten Aktionsanzeigerkombinationen den Aktionsanzeiger β4 enthält.

2.1.2 Transformationsregeln

Die Mehrtreffer-Entscheidungstabelle benötigt insbesondere für die *Regeltransformationen* andere Verfahren, da die logischen Beziehungen der Bedingungsanzeigerkombinationen vielfältiger sind als bei der Eintreffer-Entscheidungstabelle. Aus dem gleichen Grund lassen sich auch nicht die Verfahren der *Regelminimierung* unverändert übernehmen.

Da die Mehrtreffer-Entscheidungstabelle über verschiedene Formate verfügt, müssen Transformationen von einem Format in das andere ebenso möglich sein wie Transformationen der Mehrtreffertabelle in die Eintreffer-Entscheidungstabelle und vice versa.

2.1.2.1 Regeltransformationen

Syntaktische Regeltransformationen erfolgen bei Entscheidungstabellen häufig mit der Zielsetzung, die Anzahl der Entscheidungsregeln zu minimieren. Wie noch nachzuweisen sein wird, ergeben sich außerdem durchaus pragmatische Vorteile, wenn innerhalb jeder Regel auch die von "~" oder "#" verschiedenen Anzeiger minimiert werden. Damit ist jedoch in der Zielsetzung eine Analogie zum Problem der Minimierung von Schaltkreisen gegeben, so dass die in diesem Bereich zur Minimierung von Aussagenfunktionen entwickelten Verfahren auch zur Regelminimierung geeignet sein müssten.

2.1.2.1.1 Elementare Regeltransformationen

Ausgangspunkt für eine Erörterung sinnvoller Regeltransformationen bei der Mehrtreffer-Entscheidungstabelle sind die zulässigen logischen Beziehungen der Bedingungsanzeigerkombinationen zueinander. Beide Arten der Anzeigerkombinationen können in folgenden logischen Beziehungen stehen:

Identität
Inklusion
Überschneidung
Exklusion

Daraus lassen sich für die logischen Beziehungen zweier Entscheidungsregeln zueinander insgesamt 16 Fälle bilden, die zunächst durch Beispiele illustriert werden sollen.

Logische Beziehung der Bedingungsanzeigerkombinationen \ Logische Beziehung der Aktionsanzeigerkombinationen	Identität	Inklusion	Überschneidung	Exklusion
Identität	J J - - N N ======= X X X X X X	J J - - N N ======= X X X X X	J J - - N N ======= X X X X	J J - - N N ======= X X X
Inklusion	J J - J N N ======= X X X X X X	J J J J - J - J N N N N ======= X X X X X X X X X X	J J - J N N ======= X X X X	J J - J N N ======= X X X
Überschneidung	J - - J N N ======= X X X X X X	J - - J N N ======= X X X X X X X X	J - - J N N ======= X X X X	J - - J N N ======= X X X
Exklusion	J N - J N N ======= X X X X X X	J N - J N N ======= X X X X X	J N - J N N ======= X X X X	J N - J N N ======= X X X

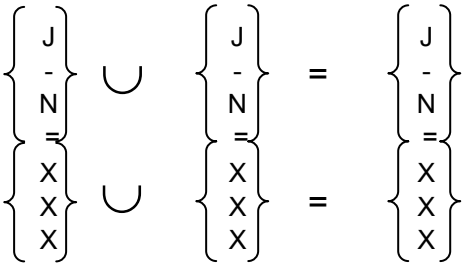
Im Folgenden sollen die bei den insgesamt 16 Fällen zulässigen Transformationen erläutert werden.

1. Identität der Bedingungsanzeigerkombinationen

   Es liegt nahe, zwei Regeln, deren Bedingungsanzeigerkombinationen identisch sind, zu vereinigen. Folgende Resultate ergeben sich in Abhängigkeit von der logischen Beziehung der Aktionsanzeigerkombinationen:

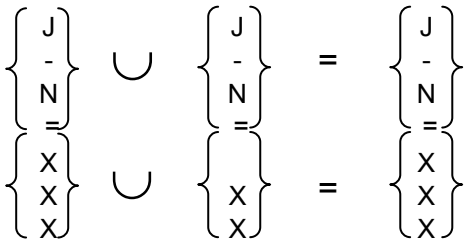
   a) Identität der Aktionsanzeigerkombinationen Beide Regeln sind formal identisch. Ergebnis der Vereinigungsoperation ist eine Regel, die zu beiden Ausgangsregeln identisch ist.

Beispiel:



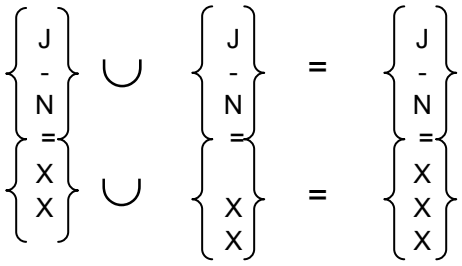
   b) Inklusion der Aktionsanzeigerkombinationen
   Die Vereinigungsoperation ergibt eine Regel, deren Bedingungsanzeigerkombination den Anzeigerkombinationen der Ausgangsregeln entspricht und deren Aktionsanzeigerkombination von der Regel mit der umfassenden Aktionsanzeigerkombination übernommen ist.

Beispiel:



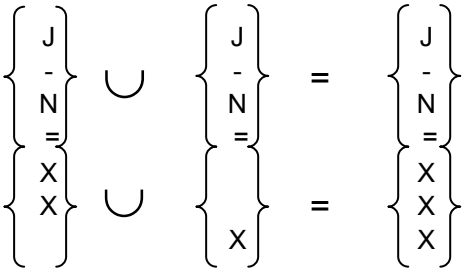
   c) Überschneidung der Aktionsanzeigerkombinationen
   Die Vereinigungsoperation ergibt eine Regel, deren Bedingungsanzeigerkombination den Anzeigerkombinationen der Ausgangsregeln entspricht und deren Aktionsanzeigerkombination sich als Erweiterungsmenge der Anzeigerkombinationen beider Regeln ergibt.

Beispiel:



d) Exklusion der Aktionsanzeigerkombinationen
Die Vereinigungsoperation ergibt eine Regel, deren Bedingungsanzeigerkombination den Anzeigerkombinationen der Ausgangsregeln entspricht und deren Aktionsanzeigerkombination als Erweiterungsmenge der Anzeigerkombination beider Regeln entsteht.

Beispiel:

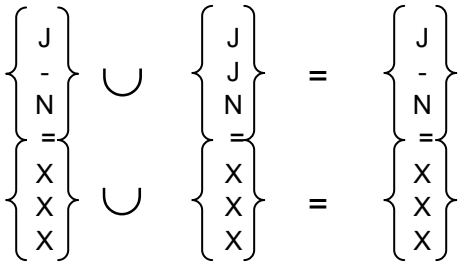


2. Inklusion der Bedingungsanzeigerkombinationen

Auch bei der Inklusion liegt es nahe zu versuchen, aus beiden Regeln durch Vereinigung eine einzige Regel zu bilden. Dieser Vereinigung sind jedoch durch die zugeordneten Aktionsanzeigerkombinationen Grenzen gesetzt. Folgende Transformationen sind zulässig:

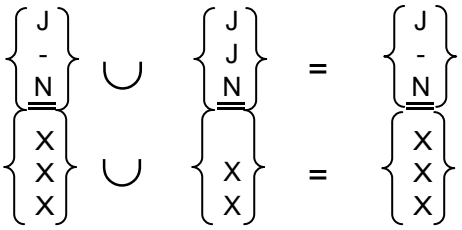
a) Identität der Aktionsanzeigerkombinationen
Da der Bedingungsanzeigerkombination der Regel, die eine Teilmenge der Bedingungsanzeigerkombination der anderen Regel darstellt, die gleiche Aktionsanzeigerkombination zugeordnet ist, kann die gesamte Regel entfallen.

Beispiel:



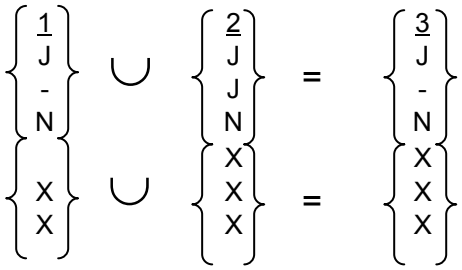
b) Inklusion der Aktionsanzeigerkombinationen
Hier lassen sich zwei Fällen unterscheiden, die bei der Transformation unterschiedlich behandelt werden müssen. Verfügt die Regel, deren Bedingungsanzeigerkombination eine Teilmenge der Bedingungsanzeigerkombination der anderen Regel darstellt, auch über eine Aktionsanzeigerkombination, die eine Teilmenge der Aktionsanzeigerkombination der anderen Regel darstellt, dann kann diese Regel eliminiert werden.

Beispiel:

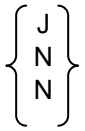


Ist jedoch der umfassenderen Bedingungsanzeigerkombination die Teilmenge der Aktionsanzeigerkombinationen zugeordnet, dann würde eine Vereinigung ein logisch falsches Ergebnis bringen.

Beweis:



Nach Vereinigung ist auch dem Komplement der Bedingungsanzeigerkombination der Regel 2 bezüglich Regel 1 die umfassendere Aktionsanzeigerkombination zugeordnet.



In der ursprünglichen Form der Regeln war diese Bedingungsanzeigerkombination jedoch nur in der Regel 1 mit der Untermenge der Aktionsanzeiger enthalten. Die Vereinigung der Regeln ist deshalb nicht zulässig.

In der zweiten Regel kann jedoch eine Überdefinition in der Aktionsanzeigerkombination beseitigt werden.

Da der Bedingungsanzeigerkombination

J
J
N

durch die Regel 1 bereits die zweite und dritte Aktion zugeordnet wurde, reicht es aus, ihr in der Regel 2 nur noch die zusätzliche erste Aktion zuzuordnen.

Für das Beispiel ergibt sich danach

Beispiel:

J	J			J	J		
J	J			-	J		
N	N			N	N		
X		äqsyn		X			
X	X			X			
	X				X		

c) Überschneidung der Aktionsanzeigerkombinationen

Die Vereinigung der Regeln ist nicht zulässig, da der Teilmenge der Bedingungsanzeiger teilweise andere Aktionsanzeiger zugeordnet sind. Aus der Regel mit der Teilmenge der Bedingungsanzeiger kann jedoch der Durchschnitt der Aktionsanzeiger eliminiert werden.

J	J			J	J		
-	J			-	J		
N	N			N	N		
	X	äqsyn			X		
X	X			X			
X	X			X			

d) Exklusion der Aktionsanzeigerkombinationen

Die Untermenge der Bedingungsanzeiger kann aus der umfassenderen Regel eliminiert werden, wenn ihr in der anderen Regel die Vereinigung der Aktionsanzeigerkombinationen zugeordnet wird.

Beispiel:

$$\left\{\begin{array}{cc} J & J \\ - & J \\ \underline{\underline{N}} & \underline{\underline{N}} \\ X & \\ X & \\ X & X \end{array}\right\} \text{ äqsyn } \left\{\begin{array}{cc} J & J \\ N & J \\ \underline{\underline{N}} & \underline{\underline{N}} \\ X & X \\ X & X \\ & X \end{array}\right\}$$

3. Überschneidung der Bedingungsanzeigerkombinationen

Bei einer Überschneidung der Bedingungsanzeigerkombinationen kann dem Durchschnitt der Bedingungsanzeigerkombinationen die Vereinigungsmenge der Aktionsanzeigerkombinationen zugeordnet werden.

Je nach logischer Beziehung der Aktionsanzeigerkombinationen ergeben sich dann unterschiedliche Auswirkungen auf die ursprünglichen Regeln.

a) Identität der Aktionsanzeigerkombinationen
Der Durchschnitt der Bedingungsanzeigerkombinationen kann aus einer der beiden Regeln eliminiert werden.

Beispiel:

$$\left\{\begin{array}{c} J \\ - \\ \underline{\underline{N}} \end{array}\right\} \cap \left\{\begin{array}{c} - \\ J \\ \underline{\underline{N}} \end{array}\right\} = \left\{\begin{array}{c} J \\ J \\ \underline{\underline{N}} \end{array}\right\}$$

$$\left\{\begin{array}{c} X \\ X \\ X \end{array}\right\} \cup \left\{\begin{array}{c} X \\ X \\ X \end{array}\right\} = \left\{\begin{array}{c} X \\ X \\ X \end{array}\right\}$$

Da

$$\left\{\begin{array}{c} J \\ - \\ N \end{array}\right\} - \left\{\begin{array}{c} J \\ J \\ N \end{array}\right\} = \left\{\begin{array}{c} J \\ N \\ N \end{array}\right\} \text{ und } \left\{\begin{array}{c} - \\ J \\ N \end{array}\right\} - \left\{\begin{array}{c} J \\ J \\ N \end{array}\right\} = \left\{\begin{array}{c} N \\ J \\ N \end{array}\right\}$$

folgt:

$$\left\{\begin{array}{cc} J & - \\ - & J \\ \underline{\underline{N}} & \underline{\underline{N}} \\ X & X \\ X & X \\ X & X \end{array}\right\} \text{ äqsyn } \left\{\begin{array}{cc} J & - \\ N & J \\ \underline{\underline{N}} & \underline{\underline{N}} \\ X & X \\ X & X \\ X & X \end{array}\right\} \text{äqsyn } \left\{\begin{array}{cc} J & N \\ - & J \\ \underline{\underline{N}} & \underline{\underline{N}} \\ X & X \\ X & X \\ X & X \end{array}\right\}$$

b) Inklusion der Aktionsanzeigerkombinationen

Da dem Durchschnitt der Bedingungsanzeigerkombinationen auch die Vereinigungsmenge der Aktionsanzeigerkombinationen zugeordnet werden muss, kann der Durchschnitt nur aus der Regel mit der umfassenderen Aktionsanzeigerkombination eliminiert werden.

Beispiel:

$$\left\{\begin{matrix} J \\ - \\ \underline{N} \\ X \\ X \\ X \end{matrix}\right\} \cap \left\{\begin{matrix} - \\ J \\ \underline{N} \\ \\ X \\ X \end{matrix}\right\} = \left\{\begin{matrix} J \\ J \\ \underline{N} \\ X \\ X \\ X \end{matrix}\right\}$$

$$\left\{\begin{matrix} J \\ - \\ \underline{N} \\ X \\ X \\ X \end{matrix}\right\} \cup \left\{\begin{matrix} - \\ J \\ \underline{N} \\ \\ X \\ X \end{matrix}\right\} = \left\{\begin{matrix} J \\ - \\ \underline{N} \\ X \\ X \\ X \end{matrix}\right\}$$

Da

$$\left\{\begin{matrix} J \\ - \\ N \end{matrix}\right\} - \left\{\begin{matrix} J \\ J \\ N \end{matrix}\right\} = \left\{\begin{matrix} J \\ N \\ N \end{matrix}\right\}$$

Folgt

$$\left\{\begin{matrix} J\ - \\ -\ J \\ \underline{N\ N} \\ X \\ X\ X \\ X\ X \end{matrix}\right\} \text{äqsyn} \left\{\begin{matrix} J\ - \\ N\ J \\ \underline{N\ N} \\ X \\ X\ X \\ X\ X \end{matrix}\right\}$$

c) Überschneidung der Aktionsanzeigerkombinationen

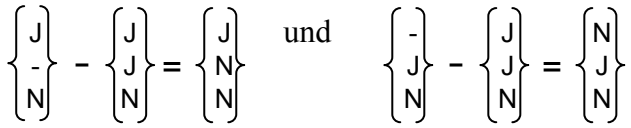
Die Bedingungsanzeigerkombination der neuen Regel, die sich aus dem Durchschnitt der Bedingungsanzeigerkombinationen und der Vereinigung der Aktionsanzeigerkombinationen der beiden ursprünglichen Regeln zusammensetzt, kann aus beiden Bedingungsanzeigerkombinationen der ursprünglichen Regeln gemeinsam entfernt werden.

Beispiel:

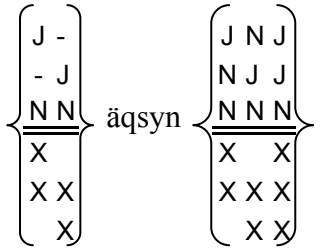
$$\left\{\begin{matrix} J \\ - \\ \underline{N} \\ X \\ X \end{matrix}\right\} \cap \left\{\begin{matrix} - \\ J \\ \underline{N} \\ \\ X \\ X \end{matrix}\right\} = \left\{\begin{matrix} J \\ J \\ \underline{N} \\ X \\ X \\ X \end{matrix}\right\}$$

$$\left\{\begin{matrix} J \\ - \\ \underline{N} \\ X \\ X \end{matrix}\right\} \cup \left\{\begin{matrix} - \\ J \\ \underline{N} \\ \\ X \\ X \end{matrix}\right\} = \left\{\begin{matrix} J \\ - \\ \underline{N} \\ X \\ X \\ X \end{matrix}\right\}$$

Da



folgt

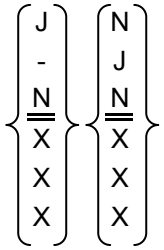


äqsyn

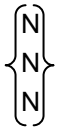
d) Exklusion der Aktionsanzeigerkombinationen
Die Möglichkeiten der Transformation entsprechen dem Fall c).

4. Exklusion der Bedingungsanzeigerkombinationen

Nur bei *Identität der Aktionsanzeigerkombinationen* können beide Regeln zu einer einzigen zusammengefasst werden, wenn zu den Bedingungsanzeigerkombinationen eine einzige syntaktisch äquivalente Kombination existiert.
Beispiel:

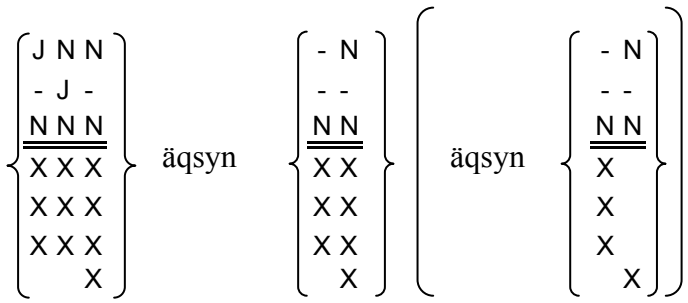


Keine Vereinigung möglich, da es zu den beiden aufgeführten Bedingungsanzeigerkombinationen keine einzige syntaktisch äquivalente Bedingungsanzeigerkombination gibt. Es fehlt die Bedingungsanzeigerkombination.
Beispiel:



Kann in der Entscheidungstabelle eine weitere Regel nachgewiesen werden, die die fehlende Bedingungsanzeigerkombination enthält und deren Aktionsanzeigerkombination nicht im Widerspruch zu der Anzeigerkombination der untersuchten Regeln steht, dann kann die Vereinigung durchgeführt werden.

Beispiel:



2.1.2.1.2 Verfahren zur kombinierten Regel – und Anzeigerminimierung

Die bekanntesten Verfahren zur Minimierung von Aussagenfunktionen sind die graphische Minimierung mit Hilfe des Karnaugh-Diagramms und das Quine-Mc-Cluskey-Verfahren. Beide Verfahren sollen auf ihre Anwendbarkeit zur Regelminimierung von Mehrtreffer-Entscheidungstabellen überprüft werden

2.1.2.1.2.1 Das Karnaugh -Diagramm

In der Theorie des Schaltkreisentwurfs gilt das Karnaugh-Diagramm als Verfahren zur graphischen Darstellung von Aussagenfunktionen und zu ihrer Vereinfachung.[1)]

Das Karnaugh -Diagramm besteht aus matrixförmig angeordneten Feldern, in denen die Wahrheitswerte der Variablen einer Aussagenfunktion eingetragen werden können. Die Größe der Feldermatrix wird durch die Anzahl der Variablen in der Aussagenfunktion bestimmt, die Zeilen und Spalten der Matrix werden systematisch mit den möglichen Wahrheitswerten versehen. Das Karnaugh -Diagramm kann insoweit als Alternative zur Wertetabelle gelten.

Auf Entscheidungstabellen mit einfachen Bedingungen, die ebenso wie einfache Aussagenvariablen nur zwei Zustände - J und N - annehmen können, ist das Karnaugh - Diagramm unmittelbar anwendbar.[2)]

In einer Mehrtreffer-Entscheidungstabelle sollen den folgenden Bedingungsanzeigerkombinationen identische Aktionsanzeigerkombinationen zugeordnet sein:

Bedingung a	J	J	J	J	J	J	J	N	N	N
Bedingung b	J	J	J	J	N	N	N	J	N	N
Bedingung c	J	J	N	N	J	N	N	N	J	N
Bedingung d	J	N	J	N	N	J	N	J	N	N

1) Vgl. Caldwell (Entwurf von Schaltkreisen), S. 122 ff.

2) Darauf hat zuerst O'Brien hingewiesen. Vgl. O'Brien (Computerizing administrative operations), S. 72.
Vgl. ferner Sprague (Storage space), S. 319; Beizer (Architecture), S. 203 ff.; Rabin (Optimal decision trees), S. 69

Diese Logik wird in einem Karnaugh -Diagramm wie folgt dargestellt:

cd \ ab	NN	NJ	JJ	JN
N N	1		1	1
N J		1	1	1
J J			1	
J N	1		1	1

Jede "1" im Diagramm zeigt an, dass die dadurch definierte Bedingungsanzeigerkombination innerhalb der untersuchten Regeln der Entscheidungstabelle vertreten ist.

Eine graphische Minimierung ist möglich, weil das Karnaugh -Diagramm folgende grundlegende Eigenschaften aufweist:[1)]

1. Jedes Feld des Diagramms ist einer ganz bestimmten Bedingungsanzeigerkombination (allgemein: Wertekombination der Variablen) zugeordnet.

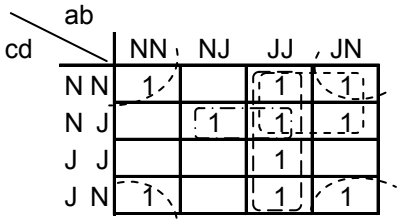
2. Beim Übergang von einem Feld zu einem benachbarten wechselt jeweils nur eine Bedingung (Variable) ihren Anzeiger (Wert). Zusätzlich zu den unmittelbar neben- oder untereinander angeordneten Feldern gelten dabei auch die beiden Felder an den Enden jeder Spalte und die beiden Felder an den Enden jeder Zeile als benachbart.

Die Bildung komplexer Regeln wird dann dadurch möglich, dass für durch "1" gekennzeichnete benachbarte Felder nur eine Regel gebildet wird, in der die unveränderten Bedingungsanzeiger erhalten bleiben und die wechselnden Anzeiger durch den Indifferenzanzeiger repräsentiert werden. Um diesen Prozess leicht zu veranschaulichen, werden benachbarte Felder durch Schleifen umrandet.

---

1) Vgl. Hoernes; Heilweil (Logik Entwurf), S. 189

In dem als Beispiel verwendeten Karnaugh -Diagramm können die Minimierungsmöglichkeiten wie folgt gekennzeichnet werden:



Es lassen sich danach folgende komplexe Regeln bilden:

	1	2	3	4
Bedingung a	-	J	J	-
Bedingung b	J	J	-	N
Bedingung c	N	-	N	-
Bedingung d	J	-	-	N

Damit ist gleichzeitig sowohl die Regelanzahl minimiert als auch die Anzahl der von "-" verschiedenen Anzeiger innerhalb der Regeln.

Wenn auch bei dem angegebenen Beispiel das graphische Minimierungsverfahren mit Hilfe des Karnaugh -Diagramms sehr eindrucksvoll erscheint, stehen seiner Verwendung innerhalb der Entscheidungstabellentechnik schwerwiegende Bedenken entgegen:

1. Der Minimierungsprozess wird recht unübersichtlich, wenn mehr als acht Variablen zu berücksichtigen sind. Entscheidungstabellen mit mehr als acht Bedingungen sind jedoch durchaus üblich.

2. Die Anschaulichkeit des graphischen Minimierungsverfahrens resultiert daraus, dass jede Variable nur zwei Zustände hat. Zwar lassen sich auch Karnaugh -Diagramme zu Variablen mit mehr als zwei Zuständen konstruieren, das graphische Minimierungsverfahren ist dann jedoch nicht mehr durchführbar, weil die visuelle Erkennung benachbarter Felder nicht mehr in allen Fällen gelingt. Das Verfahren ist deshalb bei erweiterten und gemischten Entscheidungstabellen nicht anwendbar.

2.1.2.1.2.2 Das Quine-McCluskey-Verfahren und seine Modifikation

Ein weiteres Verfahren zur Minimierung von Aussagenfunktionen, das Quine-McCluskey-Verfahren (QMC-Verfahren) [1] eignet sich besonders gut zu einer automatischen Abwicklung. Wie noch zu zeigen sein wird, lässt sich das QMC -Verfahren so modifizieren, dass auch Variablen mit mehr als zwei Zuständen behandelt werden können. Da außerdem die Anwendbarkeit des Verfahrens nicht von der Zahl der Variablen abhängig ist, erfüllt das modifizierte QMC -Verfahren alle Anforderungen, die an ein syntaktisches Verfahren zur Regelminimierung von Mehrtrefferentscheidungstabellen gerichtet werden müssen.

Das QMC -Verfahren wird in folgenden Teilschritten ausgeführt:[2]

1. Die zu behandelnde Aussagenfunktion wird in ihre alternative Normalform überführt.

2. Die Elementarkonjunktionen werden als Binärziffernkombinationen formuliert. Der Wahrheitswert "wahr" wird durch "1", der Wahrheitswert "falsch" durch "0" repräsentiert.

3. Die Binärausdrücke werden in Gruppen mit identischer Anzahl enthaltener 1-Positionen eingeordnet. Die Gruppen werden, geordnet nach zunehmender Anzahl von 1-Positionen, in einer Spalte aufgetragen.

4. Zu jedem Binärausdruck wird das dezimale Äquivalent bestimmt und zusätzlich vermerkt.

5. Binärausdrücke zweier benachbarter Gruppen werden zusammengefasst, wenn sie sich nur in einer Position durch 0 - 1 unterscheiden. Der zusammengefasste Binärausdruck erhält einen "-" zur Kennzeichnung der Position der eliminierten Variable. Die Dezimalzahlen werden auch neben dem zusammengefassten Binärausdruck vermerkt.

6. Alle zusammengefassten Ausdrücke werden wiederum gruppiert und erneut zusammengefasst.

7. Identische zusammengefasste Binärausdrücke können bis auf einen gestrichen werden.

8. Das Endergebnis setzt sich aus allen voneinander verschiedenen Binärausdrücken zusammen, die nicht bei einer folgenden Zusammenfassung berücksichtigt werden konnten.

---

1) Vgl. Caldwell (Entwurf von Schaltkreisen), S. 133 ff.
2) Vgl. Hoernes; Heilweil (Logik-Entwurf), S. 129 ff.

Beispiel: [1)]

Die Aussagenfunktion A ∧ B ∧ D ∨ A ∧ D ∨ ~ A ∧ B ∧ C ∧ D ∨ ~ A ∧ B ∧ C ~ D
soll minimiert werden.

1. Schritt: Alternative Normalform
A ∧ B ∧ C ∧ D ∨ A ∧ B ∧ ~ C ∧ D ∨ A ∧ ~ B ∧ C ∧ D ∨
A ∧ ~B ∧~ C ∧ D ∨ ~ A ∧ B ∧ C ∧ D ∨ ~ A ∧ B ∧ C ∧ ~ D

2. Schritt: Umwandlung in Binärziffernkombinationen
1111 1101 1011 1001 0111 0110

3. Schritt: Gruppierung der Binärausdrücke

0110
1001
0111
1011
1101
1111

4. Schritt: Bestimmung des dezimalen Äquivalents

6	0110
9	1001
7	0111
11	1011
13	1101
15	1111

5. Schritt: Zusammenfassung der Binärausdrücke

6	0110 *v*	6,7 011-
9	1001 *v*	9,11 10-1
7	0111 *v*	9,13 1-01
11	1011 *v*	7,15 -111
13	1101 *v*	11,15 1-11
15	1111 *v*	13,15 11-1

6. Schritt: Erneute Gruppierung und Zusammenfassung

6,7	011-	9,11,13,15	1--1
9,11	10-1 v	9,11,13,15	1--1
9,13	1-01 v		
7,15	-111		
11,15	1-11 v		
13,15	11-1 v		

7. Schritt: Prüfung auf identische zusammengefasste Ausdrücke
Einer der aus 9, 11, 13, 15 gebildeten Ausdrücke kann gestrichen werden.

8. Schritt: Ergebnis

6,7	011-
7,15	-111
9,11,13,15	1--1

In die Notation der Aussagenfunktion umgewandelt, ergibt sich

$$\sim A \wedge B \wedge C \vee B \wedge C \wedge D \vee A \wedge D.$$

Nicht immer führt das beschriebene Verfahren zu einem absoluten Minimum an Konjunktionen, so dass das Ergebnis noch auf redundante Konjunktionen überprüft werden muß.[1)] Dazu werden die nicht mehr weiter zusammenfassbaren Binärausdrücke daraufhin analysiert, welche Elementarkonjunktionen in ihnen enthalten sind. Das ist sehr einfach an den zugeordneten Dezimaläquivalenten der Elementarkonjunktionen ablesbar. Kann nun festgestellt werden, dass ein Binärausdruck nur Elementarkonjunktionen enthält, die alle auch in den anderen Binärausdrücken vertreten sind, dann kann er eliminiert werden.

Beispiel:
Als Ergebnis des 8. Schritts wurde ermittelt:

---

1) Diese Analyse wird innerhalb des Quine-McCluskey-Verfahrens mit Hilfe der Stammgruppen-Tabelle durchgeführt.
Vgl. Caldwell (Entwurf von Schaltkreisen). S. 137 ff.

6,7	011-
17,15	-111
9,11,13,15	1--1

Der zweite Binärausdruck kann eliminiert werden, da die durch das Dezimaläquivalent "7" gekennzeichnete Elementarkonjunktion im ersten Binärausdruck und die durch das Dezimaläquivalent "15" gekennzeichnete Elementarkonjunktion im dritten Binärausdruck enthalten ist. Als endgültiges Ergebnis der Minimierung ergibt sich damit in der Notation der Aussagenfunktion:

$$\sim A \wedge B \wedge C \vee A \wedge D$$

Das erläuterte Verfahren eignet sich nicht zur Minimierung der Eintreffer-Entscheidungstabelle[1], ist jedoch direkt auf die Minimierung begrenzter Mehrtreffer-Entscheidungstabellen übertragbar.[2]

Um es jedoch auch auf erweiterte und gemischte Mehrtreffertabellen anwenden zu können, muss es modifiziert werden.

Erforderlich sind

- ein anderer Gruppierungs- und Zusammenfassungsalgorithmus, der sich auch zur Behandlung von Variablen mit mehr als zwei Zuständen eignet,
- ein anderes Verfahren zur Kennzeichnung der Elementarkonjunktionen.

Das modifizierte Verfahren soll an folgendem Beispiel illustriert werden:

Für

Bedingung a mit [a] = {a1,a2,a3}
Bedingung b mit [b] ={b1,b2,b3}
Bedingung c mit [c] = {J,N}
Bedingung d mit [d] = {d1,d2,d3}

sind folgenden Regeln einer Mehrtreffer-Entscheidungstabelle identische Aktionsanzeigerkombinationen zugeordnet:

---

1) Vgl. Myers (Compiling optimized code), S. 494
2) Vgl. Strunz (Minimization), S. 4 ff.

	1	2	3	4	5	6
Bedingung a	-	a1	a2	a3	a1	a1
Bedingung b	b1	-	b1	b1	b2	b3
Bedingung c	J	N	J	J	J	-
Bedingung d	d1	d1	-	-	d1	d1

Regelanzahl und Bedingungsanzeiger sollen minimiert werden.

1. Schritt: Die Tabelle wird in ihre kanonische Normalform überführt, was erforderlich macht, dass identische elementare Anzeigerkombinationen bis auf eine eliminiert werden.

a1	a2	a3	a1	a1	a1	a2	a2	a2	a3	a3	a3	a1	a1	a1
b1	b1	b1	b1	b2	b3	b1	b1	b1	b1	b1	b1	b2	b3	b3
J	J	J	N	N	N	J	J	J	J	J	J	J	J	N
d1	d1	d1	d1	d1	d1	d1	d2	d3	d1	d2	d3	d1	d1	d1

2. Schritt: Zu jeder Bedingungsanzeigerkombination wird eine Kennzahl als Summe der Produkte eines pro Bedingungszeile gültigen Faktors mit einem dem jeweiligen Bedingungsanzeiger zugeordneten Gewicht errechnet.[1)] Der Faktor wird nach folgender Vorschrift errechnet:

- Die unterste Bedingungszeile erhält den Faktor 1.
- Die nächst höhere Bedingungszeile erhält als Faktor die Anzahl der Bedingungsanzeiger der untersten Bedingung.
- Alle anderen Bedingungen erhalten als Faktor das Produkt aus der Anzahl der Bedingungsanzeiger der darunter angeordneten Bedingungen.
  Die Bedingungsanzeigergewichtung besteht in der Zuordnung von 0, 1, 2, 3, ... zu den lexikographisch geordneten Anzeigern.

---

1) Ein ähnliches Verfahren benutzt Veinott zur Regelkennzeichnung in dem von ihm vorgeschlagenen Codierverfahren für Entscheidungstabellen. Vgl. Veinott (Programming decision tables), s. 33 f.

Es ergeben sich folgende Kennzahlen:

	Gewicht	Faktor	Kennzahlen											
			0	18	36	3	9	15	19	20	37	38	6	12
Bedingung a	a1 = 0 a2 = 1 a3 = 2	18	a1	a2	a3	a1	a1	a1	a2	a2	a3	a3	a1	a1
Bedingung b	b1 = 0 b2 = 1 b3 = 2	6	b1	b1	b1	b1	b2	b3	b1	b1	b1	b1	b2	b3
Bedingung c	J N	3	J	J	J	N	N	N	J	J	J	J	J	J
Bedingung d	d1 = 0 d2 = 1 d3 = 2	1	d1	d1	d1	d1	d1	d1	d2	d3	d2	d3	d1	d1

3. Schritt: Gruppieren der Bedingungsanzeigerkombinationen nach den Annzeigern der obersten Bedingung und Zusammenfassen der Anzeigerkombinationen aller Gruppen, die bis auf die erste Zeile identisch sind. Aus Gründen der Übersichtlichkeit und einfacheren Durchführung der Vergleichsoperationen werden von nun an die Bedingungsanzeiger nur noch durch Ziffern repräsentiert.

0	3	9	15	6	12	18	19	20	36	37	38	0 18 36
1	1	1	1	1	1	2	2	2	3	3	3	-
1	1	2	3	2	3	1	1	1	1	1	1	1
1	2	2	2	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	2	3	1	2	3	1
v						*v*			*v*			

4. Schritt: Gruppieren der Bedingungsanzeigerkombinationen nach den Anzeigern der zweiten Bedingung und Zusammenfassen der Anzeigerkombinationen.

0	3	18	19	20	36	37	38	9	6	15	12	0 3 6 9 12 15
1	1	2	2	2	3	3	3	1	1	1	1	1 1
1	1	1	1	1	1	1	1	2	2	3	3	- -
1	2	1	1	1	1	1	1	2	1	2	1	1 2
1	1	1	2	3	1	2	3	1	1	1	1	1 1
v	*v*							*v*	*v*	*v*	*v*	

5. Schritt: Gruppieren der Bedingungsanzeigerkombinationen nach den Anzeigern der dritten Bedingung und Zusammenfassen der Anzeigerkombinationen.

0	18	19	20	36	37	38	6	12	3	9	15
1	2	2	2	3	3	3	1	1	1	1	1
1	1	1	1	1	1	1	2	3	1	2	3
1	1	1	1	1	1	1	1	1	2	2	2
1	1	2	3	1	2	3	1	1	1	1	1
v							*v*	*v*	*v*	*v*	*v*

0	6	12
3	9	15
1	1	1
1	2	3
-	-	-
1	1	1

6. Schritt: Gruppieren der Bedingungsanzeigerkombinationen nach den Anzeigern der vierten Bedingung und Zusammenfassen der Anzeigerkombinationen.

0	18	36	6	12	3	9	15	19	37	20	38
1	2	3	1	1	1	1	1	2	3	2	3
1	1	1	2	3	1	2	3	1	1	1	1
1	1	1	1	1	2	2	2	1	1	1	1
1	1	1	1	1	1	1	1	2	2	3	3
v								*v*	*v*	*v*	*v*

18	36
19	37
20	38
2	3
1	1
1	1
-	-

7. Schritt: Zusammenstellen aller komplexen Bedingungsanzeigerkombinationen und Verifikation, dass alle elementaren Bedingungsanzeigerkombinationen in mindestens einer der komplexen Anzeigerkombinationen vertreten sind. Das kann durch Vergleich der Kennzahlen erfolgen.

0	0	3				18	36
18	6	9	0	6	12	19	37
36	12	15	3	9	15	20	38
-	1	1	1	1	1	2	3
1	-	-	1	2	3	1	1
1	1	2	-	-	-	1	1
1	1	1	1	1	1	-	-

8. Schritt: Gruppieren der komplexen Anzeigerkombinationen nach den Anzeigern der obersten Bedingung. Anzeigerkombinationen mit "-" bleiben unberücksichtigt. Erneute Zusammenfassung der Anzeigerkombinationen.

0	3				18	36
6	9	0	6	12	19	37
12	15	3	9	15	20	38
1	1	1	1	1	2	3
-	-	1	2	3	1	1
1	2	-	-	-	1	1
1	1	1	1	1	-	-

Keine Zusammenfassung möglich

9. Schritt: Gruppieren nach der zweiten Bedingung und Zusammenfassen.

						0
						3
						6
0		18	36			9
18	0	19	37	6	12	12
36	3	20	38	9	15	15
-	1	2	3	1	1	1
1	1	1	1	2	3	-
1	-	1	1	-	-	-
1	1	-	-	1	1	1
	v			*v*	*v*	

10. Schritt: Gruppieren nach der dritten Bedingung und Zusammenfassen.

					0
					3
					6
0	0	18	36	3	9
18	6	19	37	9	12
36	12	20	38	15	15
-	1	2	3	1	1
1	-	1	1	-	-
1	1	1	1	2	-
1	1	-	-	1	1
	v			*v*	

11. Schritt: Gruppieren nach der vierten Bedingung und Zusammenfassung.
Vorhandene Anzeigerkombinationen erlauben keine Gruppenbildung.

12. Schritt: Zusammenstellen der komplexen Anzeigerkombinationen zweiter Stufe und der komplexen Anzeigerkombinationen erster Stufe, die nicht in die Kombinationen zweiter Stufe eingegangen sind.

0	0			
3	3			
6	6			
9	9	0	18	36
12	12	18	19	37
15	15	36	20	38

1	1	-	2	3
-	-	1	1	1
-	-	1	1	1
1	1	1	-	-

13. Schritt: Identische Anzeigerkombinationen werden bis auf eine gelöscht.
Weitere Zusammenfassungen sind nicht mehr möglich. Das Ergebnis lautet:

0			
3			
6			
9	0	18	36
12	18	19	37
15	36	20	38

1	-	2	3
-	1	1	1
-	1	1	1
1	1	-	-

14. Schritt: Anhand der Kennzahlen wird geprüft, ob redundante Anzeigerkombinationen enthalten sind. Dabei wird festgestellt, dass die zweite Anzeigerkombination eliminiert werden kann, da die darin enthaltenen elementaren Anzeigerkombinationen bereits durch die anderen komplexen Anzeigerkombinationen repräsentiert werden.
Als endgültiges Ergebnis können dann nach Einsetzen der ursprünglichen Bedingungsanzeiger folgende Regeln ausgewiesen werden.

	1	2	3
Bedingung a	a1	a2	a3
Bedingung b	-	b1	b1
Bedingung c	-	J	J
Bedingung d	d1	-	-

2.1.2.2 Formattransformationen

Charakteristisch für das *Entscheidungsmatrixformat* der Mehrtreffer -Entscheidungstabelle ist der absolute Verzicht auf die explizite Abbildung von Aktionsfolgen. Vielmehr beziehen sich alle aufgeführten Bedingungsanzeigerkombinationen immer nur auf eine einzige Aktion.

Das *Standardformat* der Mehrtreffer -Entscheidungstabelle dagegen erlaubt auch die Zuordnung von Aktionsanzeigerkombinationen zu Bedingungsanzeigerkombinationen. Unter Anwendung der im vorigen Abschnitt behandelten Regeltransformationen lassen sich die beiden Formate gegenseitig ineinander überführen.

Transformation des Standardformats in das Entscheidungsmatrixformat

Diese Transformation kann in drei Teil-Schritten abgewickelt werden:

1. Zusammenstellen der Bedingungsanzeigerkombinationen für jede Aktion. Komplexe Aktionen müssen in einfache Aktionen umgewandelt werden;

3. Formatgerechte Anordnung der syntaktischen Einheiten.

Beispiel:

Für

Bedingung a mit [a] = {al, a2, a3,}
Bedingung b mit [b] = {J, N}
Bedingung e mit [e] = {el, e2}
Bedingung d mit [d] = {J, N }

und

Aktion α mit [α] = {X}
Aktion β mit [β] = {X}
Aktion γ mit [γ] = {X}
Aktion δ mit [δ] = {δ1,δ2,δ3}

ist folgende Mehrtreffer -Entscheidungstabelle im Standardformat entwickelt worden:

		1	2	3	4	5	6	7
B1	Bedingung a	a1	-	a2	a3	-	a3	a2
B2	Bedingung b	J	N	-	N	-	N	-
B3	Bedingung c	-	c2	c1	c2	-	c2	c2
B4	Bedingung d	-	-	-	J	N	J	
A1	Aktion α	X		X		X	X	X
A2	Aktion β	X	X		X			
A3	Aktion γ		X			X		
A4	Aktion δ	δ3		δ2	δ1			δ2

1. und 2. Schritt: Zusammenstellen der Bedingungsanzeigerkombinationen für jede Aktion und Minimierung der Kombinationen durch Anwendung geeigneter Regeltransformationen

a)

	Aktion W				
B1	a1	a2	-	a3	a2
B2	J	-	-	N	-
B3	-	c1	-	c2	c2
B4	-	-	N	J	-

äqsyn

	Aktion W			
B1	a1	a2	-	a3
B2	J	-	-	N
B3	-	-	-	c2
B4	-	-	N	J

b)

	Aktion X		
B1	a1	-	a3
B2	J	N	N
B3	-	c2	c2
B4	-	-	J

äqsyn

	Aktion X	
B1	a1	-
B2	J	N
B3	-	c2
B4	-	-

c)

	Aktion Y
B1	-
B2	N
B3	c2
B4	-
	-
	-
	-
	N

d)

	Aktion $\delta = \delta 1$
B1	a3
B2	N
B3	c2
B4	J

e)

	Aktion $\delta = \delta 2$	
B1	a2	a2
B2	-	-
B3	c1	c2
B4	-	-

äqsyn

	Aktion $\delta = \delta 2$
B1	a2
B2	-
B3	-
B4	-

f)

	Aktion $\delta = \delta 3$
B1	a1
B2	J
B3	-
B4	-

3. Schritt: Formatgerechte Anordnung

	Aktion α				Aktion β		Aktion γ		Aktion δ=δ1	Aktion δ=δ2	Aktion δ=δ3
Bedingung a	a1	a2		a3	a1	-	-	-	a3	a2	a1
Bedingung b	J	-		N	J	N	N	-	N	-	J
Bedingung c	-	-		c2	-	c2	c2	-	c2	-	-
Bedingung d	-	-	N	J	-	-	-	N	J	-	-

Transformation des Entscheidungsmatrixformats in das Standardformat

Zur Umwandlung einer Entscheidungsmatrix in das Standardformat werden folgende Schritte ausgeführt:

1. Formatgerechte Anordnung von Bedingungen und Aktionen. Dabei können bei Erfüllung der Voraussetzungen komplexe Aktionen gebildet werden.

2. Aufsuchen identischer Bedingungsanzeigerkombinationen bei unterschiedlichen Aktionen. Mehrere identische Bedingungsanzeigerkombinationen können zu einer Regel zusammengesetzt werden, in der der Bedingungsanzeigerkombination eine Aktionsanzeigerkombination zugeordnet wird, die alle Aktionen repräsentiert, denen die Bedingungsanzeigerkombination in der Matrix zugeordnet war.

3. Bedingungsanzeigerkombinationen, die nur einer Aktion zugeordnet sind, werden zusammen mit dieser zu einer Regel zusammengefügt.

4. Nachdem alle Regeln gebildet worden sind, kann versucht werden, die Regelanzahl durch Anwendung zugelassener Regeltransformationen zu reduzieren.

   Beispiel:
   Die als Ergebnis der vorhergehenden Formattransformation entstandene Entscheidungsmatrix wird wieder in das Standardformat zurückverwandelt.

Es entsteht folgende Mehrtreffer-Entscheidungstabelle: [1)]

		1	2	3	4	5
B1	Bedingung a	a1	a2	-	a3	-
B2	Bedingung b	J	-	-	N	N
B3	Bedingung c	-	-	-	c2	c2
B4	Bedingung d	-	-	N	J	-
A1	Aktion α	X	X	X	X	
A2	Aktion β	X				X
A3	Aktion γ			X		X
A4	Aktion δ	δ3	δ2		δ1	

Diese Tabelle ist zwar geringfügig von der ursprünglich als Beispiel aufgeführten Tabelle verschieden, zu dieser jedoch syntaktisch äquivalent.

2.1.2.3 Transformationen zwischen Eintreffer-Entscheidungstabelle und Mehrtreffer-Entscheidungstabelle

Eine Eintreffer-Entscheidungstabelle lässt sich sehr einfach in eine Entscheidungsmatrix verwandeln. Zu jeder Aktion werden die Bedingungsanzeigerkombinationen der Regel übernommen, in denen die betreffende Aktion enthalten ist. Alle der gleichen Aktion zugeordneten Bedingungsanzeigerkombinationen können nach den erläuterten Verfahren minimiert werden.

Soll die Mehrtreffer-Entscheidungstabelle im Standardformat entstehen, dann ist dennoch zu empfehlen, Regelerstellung und- transformation zunächst für jede Aktion isoliert vorzunehmen und erst abschließend identische Bedingungsanzeigerkombinationen bei mehreren Aktionen zu einer einzigen Regel zusammenzufassen. Alle zur Umwandlung einer Eintreffer-Entscheidungstabelle in eine Mehrtreffer-Entscheidungstabelle erforderlichen Schritte sind auch Bestandteil der bereits erläuterten Formattransformationen zwischen dem Entscheidungsmatrixformat und dem Standardformat der Mehrtreffer-Entscheidungstabelle. Auf eine eingehende Erläuterung kann deshalb verzichtet werden.

Mehrere Transformationsverfahren stehen zur Umwandlung einer Mehrtreffer-Entscheidungstabelle in die Eintreffer-Entscheidungstabelle zur Verfügung.

---

1) Es wird darauf verzichtet, die Transformationsoperationen im Einzelnen zu erläutern, da sie leicht nachvollziehbar sind.

Das *erste Verfahren* benutzt die Normalform als Zwischenstufe der Transformation. Aus den Bedingungen und Bedingungsanzeigern der Mehrtreffer–Entscheidungstabelle wird die obere Hälfte einer Entscheidungstabelle in kanonischer Normalform gebildet. Die Bedingungsanzeigerkombination k_i^m jeder Regel der Mehrtreffer-Entscheidungstabelle wird mit jeder elementaren Bedingungsanzeigerkombination k_j^n der Normalform-Tabelle verglichen. Dabei können folgende Beziehungen auftreten:

$$k_i^m \cap k_j^n = \varnothing$$
$$k_i^m \supseteq k_j^n$$

Da die Normalform-Tabelle nur elementare Bedingungsanzeigerkombinationen enthält, sind die Beziehungen $k_i^m \cap k_j^n = \varnothing$ und $k_i^m \subset k_j^n$ ausgeschlossen.

Die Übertragung aller Aktionsanzeigerkombinationen $\partial\ell_i^m$ der Regeln der Mehrtreffer-Entscheidungstabelle erfolgt dann nach folgender Vorschrift:

1. Bei $k_i^m \cap k_j^n = \varnothing$ keine Übertragung der Aktionsanzeiger,
   bei $k_i^m \supseteq k_j^n$ Zuordnung der Aktionsanzeigerkombination $\partial\ell_i^m$ von k_i^m zu k_j^n.
   Falls k_j^n bereits irgendeine Aktionsanzeigerkombination $\partial\ell_k$ zugeordnet ist, dann Zuordnen des Ergebnisses der Vereinigungsoperation

$$\partial\ell_i^m \cup \partial\ell_k .$$

   Nach vollständiger Übertragung der Regeln der Mehrtreffertabelle in die Normalform-Tabelle werden mit Aktionsanzeigerkombinationen versehene Regeln der Normalform-Tabelle minimiert, die restlichen Regeln können eliminiert werden.

Das beschriebene Verfahren ist einfach durchführbar und bei manueller Ausführung wenig fehleranfällig. Die Notwendigkeit zur Erstellung einer Normalform-Tabelle begrenzt jedoch seine Einsetzbarkeit. Mehrtreffer–Entscheidungstabelle mit vielen Bedingungen und/oder einer größeren Anzahl von Bedingungsanzeigern pro Bedingung sind nach der beschriebenen Methode nicht mehr transformierbar.

Ein *anderes Verfahren* verzichtet auf die Normalform-Tabelle als Zwischenstufe. Alle Regeln der Mehrtreffer-Tabelle werden durch Umwandlung komplexer Bedingungsanzeigerkombinationen in ihre elementaren Äquivalente in einfache Entscheidungsregeln transformiert. Alle so entstandenen Regeln werden untereinander verglichen und nach folgender Vorschrift behandelt:

- Bei Identität von Bedingungsanzeigerkombinationen und Verschiedenheit der Aktionsanzeigerkombinationen erhält eine Regel die Vereinigungsmenge der

Aktionsanzeigerkombinationen zugeordnet, die anderen Regeln werden eliminiert.

Bei Exklusion der Bedingungsanzeigerkombinationen erfolgt keine Veränderung.

Sind auf diese Weise alle Identitäten im Bedingungsanzeigerteil beseitigt, können die verbliebenen Regeln wieder nach den Vorschriften der Minimierung von Entscheidungsregeln der Eintreffer-Entscheidungstabelle reduziert werden. Dieses Verfahren ist in der Regel dann vorteilhafter, wenn in der Mehrtreffer-Entscheidungstabelle nur ein kleinerer Teil der formal möglichen elementaren Bedingungsanzeigerkombinationen repräsentiert ist.

Das *dritte Verfahren* schließlich versucht, die im Allgemeinen recht aufwendigen Auflösungen komplexer Regeln der Mehrtreffer–Entscheidungstabelle in einfache Regeln auf ein Minimum zu beschränken. Die Regeln der Mehrtreffer-Entscheidungstabelle werden paarweise untereinander verglichen und ihre Beziehung im Bedingungsanzeigerteil festgestellt:

Seien k_i und k_j. die Bedingungsanzeigerkombinationen unterschiedlicher Regeln der Menge R^x einer Mehrtreffer-Entscheidungstabelle, dann sind folgende Relationen möglich:

$k_i \cap k_j = \phi$	Exklusion
$k_i = k_j$	Identität
$k_i \subset k_j$	Inklusion
$k_i \cap k_j \neq \phi$	Überschneidung

Bei *Exklusion* können beide Regeln unverändert bleiben, da die Forderung der Eintreffer-Entscheidungstabelle nach disjunkten Bedingungsanzeigerkombinationen bereits erfüllt ist.

Bei *Identität* erhalt eine Regel die Vereinigungsmenge der Aktionsanzeigerkombinationen zugeordnet, die andere Regel wird eliminiert.

Bei *Inklusion* sind in Anhängigkeit von den zugeordneten Aktionsanzeigerkombinationen $\partial\ell_i$ und $\partial\ell_j$ zwei Fälle zu unterscheiden.

(1) $k_i \subset k_j$ mit $\partial\ell_i \subseteq \partial\ell_j$

(2) $k_i \subset k_j$ mit $\partial\ell_i \supset \partial\ell_j$

Im Fall (1) kann die aus k_i und $\partial\ell_i$ bestehende Regel eliminiert werden

Im Fall (2) wird durch

$$k_j - k_i = k_j'$$

das Komplement zu k_j gebildet und in der betreffenden Regel k_j durch k_j' ersetzt. Die Bildung des Komplements kann eine Auflösung der Regel erforderlich machen.

Beispiel:

i	j
J	J
J	-
J	-
X	X
X	X
X	

Da

$$\begin{Bmatrix} J \\ - \\ - \end{Bmatrix} - \begin{Bmatrix} J \\ J \\ J \end{Bmatrix} = \begin{Bmatrix} J & J \\ J & N \\ N & - \end{Bmatrix}$$

ergibt sich

i		j´
J	J	J
J	J	N
J	N	-
X	X	X
X	X	X
X		

Bei der *Überschneidung* der Bedingungsanzeigerkombinationen lassen sich in Abhängigkeit von der Beziehung der Aktionsanzeigerkombinationen zueinander drei Transformationsregeln formulieren.

1. Bei *identischen* Aktionsanzeigerkombinationen wird der Durchschnitt k^d aus den Bedingungsanzeigerkombinationen gebildet und eine der beiden ursprünglichen Bedingungsanzeigerkombinationen durch das Komplement ersetzt, das sich durch Vermindern der Bedingungsanzeigerkombination um den Durchschnitt k^d ergibt.

Für $R^x = \{(k_i;\partial\ell_i),(k_j;\partial\ell_j)\}$ gilt $k_i \cap k_j \neq \varnothing$ und $\partial\ell_i = \partial\ell_j$.
Da $k_i \cap k_j = k^d$,
folgt $k_i - k^d = k_i'$
und $k_j - k^d = k_j'$.

Es ergibt sich die neue Menge R^y zu

$R^y = \{(k_i';\partial\ell_i)(k_j;\partial\ell_j)\}$ oder $\{(k_i;\partial\ell_i),(k_j';\partial\ell_j)\}$.

Beispiel:

J	-		J	-		J	N		J	N	
-	N	äqsyn	J	N	ODER	-	N		-	N	
J	J		J	J		J	J		J	J	
X	X		X	X		X	X		X	X	
X	X		X	X		X	X		X	X	
X	X		X	X		X	X		X	X	

2. Bei *Inklusion* der Aktionsanzeigerkombinationen wird der Durchschnitt der Bedingungsanzeigerkombinationen von der Bedingungsanzeigerkombination abgezogen, der die Teilmenge der Aktionsanzeiger zugeordnet ist.

Für $R^x = \{(k_i;\partial\ell_i),(k_j;\partial\ell_j)\}$ gilt $k_i \cap k_j \neq \varnothing$ und $\partial\ell_i = \partial\ell_j$.
Da $k_i \cap k_j = k^d$,
folgt $k_i - k^d = k_i'$

Es ergibt sich
$R^y = \{(k_i';\partial\ell_i)(k_j;\partial\ell_j)\}$

Beispiel:

J	-		J	-	
-	N	äqsyn	J	N	
J	J		J	J	
	X			X	
X	X		X	X	
X	X		X	X	

3. Bei *disjunkten* oder sich *überschneidenden* Aktionsanzeigerkombinationen wird eine neue Regel aus dem Durchschnitt der Bedingungsanzeigerkombinationen k^d und der Vereinigungsmenge der Aktionsanzeigerkombinationen $\partial\ell^V$ gebildet. und die Bedingungsanzeigerkombinationen der ursprünglichen Regeln werden durch ihr Komplement ersetzt. das sich aus der Verminderung um den Durchschnitt ergibt.

   Für $R^X = \{(k_i; \partial\ell_i), (k_j; \partial\ell_j)\}$ gilt $k_i \cap k_j \neq \varnothing$ und $\partial\ell_i = \partial\ell_j$.
   und $\partial\ell_i \cap \partial\ell_j = \varnothing$
   oder $\partial\ell_i \cap \partial\ell_j \neq \varnothing$

   Da $k_i \cap k_j = k^d$,
   folgt $k_i - k^d = k_j$
   $k_j - k^d = k'_j$;
   Außerdem $\partial\ell_i \cup \partial\ell_j = \partial\ell^V$.

   Daraus ergibt sich
   $R^y = \{(k^d; \partial\ell^V), (k'_i; \partial\ell_i), (k'_j; \partial\ell_j)\}$.

Die Regeltransformationen sind beendet. wenn alle Bedingungsanzeigerkombinationen der verbliebenen Regeln zueinander disjunkt sind. Eine abschließende Minimierung kann die Anzahl der Regeln noch reduzieren.

Das beschriebene Verfahren eignet sich für die manuelle Transformation von Mehrtreffer-Entscheidungstabellen ebenso wie für die automatische Transformation.

2.1.3 Syntaxprüfungen

Ein Pendant zur Eindeutigkeitsprüfung der Eintreffer-Entscheidungstabelle ist für die Mehrtreffer-Entscheidungstabelle nicht notwendig. Da zwischen den Bedingungsanzeigerkombinationen der Regeln alle möglichen logischen Beziehungen erlaubt sind – Exklusion, Überschneidung, Inklusion und Identität- besteht keine Veranlassung, formale Mehrdeutigkeiten zu diagnostizieren und zu beseitigen.

Eine Syntaxprüfung zur Kontrolle der Vollständigkeit im Bereich der Entscheidungsregeln ist jedoch bei der Mehrtreffertabelle möglich und auch notwendig. Die Vollständigkeitsprüfung soll

die Bestätigung dafür erbringen, dass alle elementaren Bedingungsanzeigerkombinationen durch mindestens eine Regel der Mehrtreffertabelle repräsentiert werden. Es ist leicht einzusehen, dass die zur Vollständigkeitsprüfung der Eintreffer-Entscheidungstabelle entwickelten rechnerischen Verfahren nicht unmittelbar angewendet werden können. Da elementare Bedingungsanzeigerkombinationen durch verschieden viele Entscheidungsregeln abgedeckt sein·können, ist ihre Mehrfachzählung unvermeidlich und damit das Ergebnis der Vollständigkeitsprüfung unbrauchbar.

Die Vollständigkeitsprüfung mit Hilfe einer äquivalenten Normalform-Tabelle wäre dagegen möglich. Jede elementare Bedingungsanzeigerkombination k_i^n der Normalform-Tabelle wird mit allen Bedingungsanzeigerkombinationen k_j^m der Regeln der Mehrtreffer-Entscheidungstabelle verglichen. Wird beim Vergleich irgendeines k_i^n mit irgendeinem k_j^m die Beziehung $k_i^n \subseteq k_j^m$ festgestellt, ist der Vollständigkeitsnachweis in Bezug auf die elementare Bedingungsanzeigerkombination k_i^n erbracht. Andernfalls wird die betreffende Anzeigerkombination als fehlend gekennzeichnet.

Dieses Verfahren, so einleuchtend es auch erscheinen mag, wird in vielen Fällen schon deshalb nicht anwendbar sein, weil der Aufwand zur Erstellung der Normalform-Tabelle zu hoch ist.

Günstiger erscheint dagegen ein Verfahren, das die rechnerische Vollständigkeitsprüfung der Eintreffer-Entscheidungstabelle auch zur Prüfung der Mehrtreffer-Entscheidungstabelle anwendbar macht. Die Mehrtreffer-Entscheidungstabelle muss dazu jedoch partiell in eine Eintreffer-Tabelle transformiert werden. Nur die *Bedingungsanzeigerkombinationen* der Regeln der Mehrtreffer-Entscheidungstabelle werden untereinander verglichen und wie folgt transformiert:

Seien k_i und k_j zwei beliebige, unterschiedliche Bedingungsanzeigerkombinationen in der Mehrtreffer-Entscheidungstabelle, dann

- bleiben k_i und k_j unverändert bei $k_i \cap k_j = \varnothing$,
- wird k_i eliminiert bei $k_i \subseteq k_j$,
- wird der Durchschnitt aus k_i und k_j gebildet und aus k_i oder k_j eliminiert bei $k_i \cap k_j \neq \varnothing$,

Wenn alle Bedingungsanzeigerkombinationen logisch disjunkt sind, kann die rechnerische Vollständigkeitsprüfung durchgeführt werden. Fehlende Bedingungsanzeigerkombinationen können nach dem ebenfalls bereits beschriebenen Verfahren zur zeilenweisen Ermittlung fehlender Bedingungsanzeigerkombinationen ermittelt werden.

2.2 Semantik der Mehrtreffer –Entscheidungstabelle

Auch bei der Erörterung der Semantik der Mehrtreffer-Entscheidungstabelle sollen nur Abweichungen von der Eintreffer-Entscheidungstabelle diskutiert werden.

2.2.1 Abbildungsfunktionen der Mehrtreffer-Entscheidungstabelle

Während innerhalb einer semantisch vollständigen und redundanzfreien Eintreffer-Entscheidungstabelle für jede Bedingungskonstellation der Realität genau eine Entscheidungsregel als zutreffend lokalisiert werden kann, können bei der Mehrtreffer-Entscheidungstabelle auch mehrere Entscheidungsregeln zutreffen. Die logische Beziehung der Entscheidungsregeln zueinander ist das inklusive ODER im Gegensatz zum exklusiven ODER der Eintreffer-Entscheidungstabelle. Das ergibt sich schlüssig aus der Art der logischen Beziehungen, die die Bedingungsanzeigerkombinationen der Regeln zueinander haben dürfen. Neben der Exklusion sind das auch Inklusion, Überschneidung und Identität.

Dennoch bleibt die Abbildung eindeutig, da folgende semantische Regel die Art der Ausführung der Aktionen unmissverständlich definiert:

Vorschrift: Treffen bei einer realen Bedingungskonstellation mehrere Regeln einer Mehrtreffer-Entscheidungstabelle zu, dann werden die Aktionsanzeigerkombinationen vereinigt. Das Ergebnis der Vereinigungsoperation bildet die auszuführende Aktionsfolge ab.[1)]

2.2.2 Verifikation der Mehrtreffer-Entscheidungstabelle

Da es nach den für den Aufbau der Mehrtreffer-Entscheidungstabelle geltenden Regeln keine Basis mehr für Mehrdeutigkeiten im Bereich der Entscheidungsregeln gibt, verbleibt als Aufgabe für die semantische Verifikation, die vollständige Abbildung von Bedingungskonstellationen und Aktionsfolgen nachzuprüfen.

---

1) Auch andere semantische Regeln der Aktionsauswertung wären möglich. Die angegebene Regel dürfte jedoch den Anforderungen der Praxis am besten gerecht werden.

2.2.2.1 Vollständigkeit bei der Abbildung der Bedingungskonstellationen

Ebenso wie bei der Prüfung der Eintreffer-Entscheidungstabelle wird bei der semantischen Vollständigkeitsprüfung der Mehrtreffer-Tabelle zunächst überprüft, inwieweit die Tabelle so gestaltet ist, dass die Ergebnisse der syntaktischen Vollständigkeitsprüfung semantisch relevant sind.

Sofern alle Bedingungsanzeiger jeder Bedingung logisch disjunkte Bedingungszustände vollständig abbilden und die Kombinierbarkeit der Bedingungsanzeiger nicht durch logische Abhängigkeiten der Bedingungen eingeschränkt ist, haben die Ergebnisse der syntaktischen Vollständigkeitsprüfung unmittelbare semantische Relevanz.

Sind die erwähnten Prämissen nicht erfüllt, müssen die syntaktischen Auswirkungen der spezifischen Überlappung der Bedingungszustände und/ oder der Bedingungsabhängigkeiten bei der Interpretation der Ergebnisse der syntaktischen Vollständigkeitsprüfung entsprechend berücksichtigt werden.

2.2.2.2 Vollständigkeit bei der Abbildung der Aktionsfolgen

Ein wesentliches Merkmal der Mehrtreffer-Entscheidungstabelle ist, dass die abgebildeten Aktionsfolgen nicht immer unmittelbar ablesbar sind, sondern sich auch erst bei der dynamischen Ausführung der Tabelle in Abhängigkeit von der realen Bedingungskonstellation als Vereinigungsmenge der innerhalb der zutreffenden Regeln abgebildeten Aktionen ergeben. Daher entfällt die Möglichkeit, durch einfaches Ablesen Unvollständigkeiten bei den definierten Aktionsfolgen zu erkennen.

Eine Bestimmung der abgebildeten Aktionsfolgen ist jedoch dadurch möglich, dass zunächst analysiert wird, welche Regeln gemeinsam zutreffen können und in allen Fällen gemeinsamen Zutreffens die Vereinigungsmenge der Aktionen gebildet wird.

Die logischen Beziehungen der Bedingungsanzeigerkombinationen umfassen bekanntlich Exklusion, Überschneidung, Inklusion und Identität. In allen Fällen, in denen nicht Exklusion (semantische Exklusion) erkannt wird, müssen die definierten Aktionen vereinigt werden. Die Anzahl der Regelgruppen, die überprüft werden müssen, steigt jedoch mit der Anzahl der in der Tabelle vorhandenen Regeln stark an. Beispielsweise können bei 6 Regeln bereits 57 Gruppen gebildet werden, bei 7 Regeln 120 und bei 8 Regeln 247. Allgemein ergibt sich die Anzahl G zu überprüfender Regelgruppen für eine Tabelle mit n Regeln aus der Formel

$$G = 2^n - n - 1 \qquad \text{mit } n \geq 2 .$$

Nur bei relativ kleinen Mehrtreffer-Entscheidungstabellen wird deshalb der Aufwand zur Bildung aller gemeinsamen Aktionsfolgen vertretbar sein. Bei größeren Entscheidungstabellen kann diese Analyse nur maschinell vorgenommen werden.

Ergebnis der Vollständigkeitsprüfung ist eine explizite Abbildung aller unterschiedlichen Aktionsfolgen, die sich aufgrund des gemeinsamen Zutreffens von Regeln ergeben können. Diese bilden eine Teilmenge der maximalen Anzahl von Aktionsfolgen, die sich syntaktisch als Produkt der Anzahl der Aktionsanzeiger (einschließlich Ausschlußanzeiger) jeder in der Tabelle abgebildeten Aktion ergibt.

Jede durch Regelkombination entstehende Aktionsfolge kann abschließend auf Vollständigkeit überprüft werden.

2.2.2.3 Widerspruchsfreiheit der Aktionsfolgen

Die bei Mehrtreffer-Entscheidungstabellen mögliche Aufteilung der Abbildung auszuführender Aktionsfolgen auf mehrere Regeln eröffnet ein Problem, das in dieser Deutlichkeit bei der Eintreffertabelle unbekannt ist: die Gefahr logischer Widersprüche innerhalb vereinigter Aktionsfolgen.

		1	2
B1	Betrag >100	J	-
B2	SUMME = 0	-	N
A1	Addiere Betrag zu Summe	X	
A2	Gehe nach Tabelle	5	6

Die Tabelle bildet die Aktionsfolge für folgende Bedingungskonstellation nicht widerspruchsfrei ab:

$$\left\{ \begin{matrix} J \\ N \end{matrix} \right\}$$

Es ist nicht klar definiert, ob Tabelle 5 oder Tabelle 6 als Folgetabelle ausgeführt werden soll.

Die Analyse solcher Widersprüche kann nicht immer über die Bildung aller semantisch relevanten Aktionsfolgen geschehen, da diese, wie nachgewiesen wurde, nur bei relativ kleinen Tabellen zumutbar ist. Es wird vielmehr empfohlen, mit der Analyse bei der Basisfolge der Aktionen der Entscheidungstabelle zu beginnen und zunächst die Quellen möglicher Widersprüche zu bestimmen. Bei jeder komplexen Aktion wird geprüft, ob ihre unterschiedlichen Ausprägungen zueinander im Widerspruch stehen, wie beispielsweise bei der komplexen Aktion

Gehe nach Tabelle: 1,2,3,4,5.

Auch einfache Aktionen werden daraufhin überprüft, ob sie zueinander widersprüchlich sind. Beispielsweise besteht zwischen den folgenden beiden Aktionen ein klar erkennbarer logischer Widerspruch:

Setze A auf Null
Berechne SUMME = $\underline{\text{SUMME}}$
$\qquad\qquad\qquad\qquad\qquad\qquad\quad$ A

Nach der Ermittlung aller Widersprüche werden gezielt Regeln ermittelt, deren gemeinsam abgebildete Aktionsfolgen zu diesen Widersprüchen führen würden. Nur die Bedingungsanzeigerkombinationen dieser Regeln müssen nun daraufhin überprüft werden, ob sie logisch disjunkt sind.

2.2.3 Minimierung der Mehrtreffer-Entscheidungstabelle

Die Minimierung von *Bedingungen und Aktionen* erfolgt bei Mehrtreffer - Entscheidungstabellen nach den auch für die Eintreffertabelle gültigen Regeln.

Die Minimierung der *Entscheidungsregeln* wird bei der Mehrtreffer -Entscheidungstabelle sinnvollerweise mit einer Minimierung der von "~" verschiedenen Bedingungsanzeiger verbunden. Dadurch wird sichergestellt, dass alle Bedingungen oder Bedingungszustände, die auf die abgebildete Aktion oder Aktionsfolge keinen Einfluss ausüben, auch optisch als indifferent erkennbar sind. Verfahren zur kombinierten Regel- und Anzeigerminimierung auf syntaktischer Grundlage wurden bereits erläutert. Nur unter den bekannten Prämissen - logisch disjunkte Bedingunganzeiger und keine logische Interdependenz der Bedingungen - erbringen diese Verfahren auch semantisch korrekte Ergebnisse.

Insbesondere bei Interdependenz der Bedingungen kann bei ausschließlicher Anwendung syntaktischer Verfahren ein Regel- und Anzeigerminimum nicht erzielt werden, da die Minimierung in der Regel auf der Basis der

logisch zulässigen elementaren Bedingungsanzeigerkombinationen erfolgt. Auch bei logischer Interdependenz der Bedingungen kann jedoch eine Übereinstimmung von syntaktischem und semantischem Regel- und Anzeigerminimum erzielt werden, wenn im Rahmen einer semantischen Analyse die logisch unzulässigen elementaren Bedingungsanzeigerkombinationen bestimmt und zur beliebigen Kombination mit vorhandenen Regeln freigegeben werden.[1)]

In einer Mehrtreffer-Entscheidungstabelle führen vier Regeln zu einer identischen Aktionsfolge. Ihre Bedingungsanzeigerkombinationen sind

$$\left\{\begin{array}{cccc} J & J & J & J \\ - & N & J & N \\ J & J & N & N \\ J & - & N & J \end{array}\right\}$$

Die Anwendung des Quine-McCluskey-Verfahrens bringt eine Reduzierung auf drei Regeln mit den Anzeigerkombinationen

$$\left\{\begin{array}{ccc} J & J & J \\ - & J & N \\ J & - & N \\ - & N & J \end{array}\right\}$$

Angenommen, im Rahmen einer semantischen Analyse wird eine logische Abhängigkeit der Bedingungen festgestellt, aus der sich die folgenden, logisch unzulässigen Bedingungskonstellationen ableiten lassen:

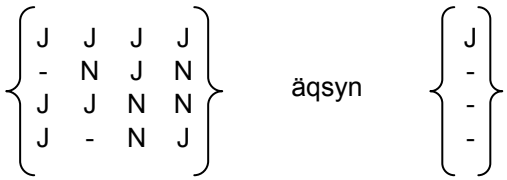
$$\left\{\begin{array}{cc} J & J \\ J & N \\ N & N \\ J & N \end{array}\right\}$$

Da diese Bedingungskonstellationen nie Bestandteil des realen Entscheidungsfeldes sein werden, kann ihre Einbeziehung in die Abbildung des Entscheidungsfeldes auch nicht schaden. Werden die beiden Anzeigerkombinationen zu den ursprünglichen vier Regeln ergänzt und in die Minimierung mit Hilfe des Quine-McCluskey-Verfahrens eingezogen, dann ergibt

---

1) Darauf hat zuerst O'Brien hingewiesen, der dies sogar zur Regelminimierung der Eintreffer-Entscheidungstabelle empfahl. Vgl. O'Brien (Computerizing administrative operations), 5.77

sich



2.2.4 Erstellung der Mehrtreffer-Entscheidungstabelle

Die zur Ermittlung von Bedingungen und Aktionen einer Eintreffer -Entscheidungstabelle vorgestellten Verfahren lassen sich ohne Einschränkung auch im Rahmen der Erstellung einer Mehrtreffertabelle anwenden. Unterschiede ergeben sich jedoch bei der Regelentwicklung. Alle systematischen Verfahren zur Regelentwicklung bei Eintreffer -Entscheidungstabellen orientieren sich an den elementaren Bedingungskonstellationen. Sie stellen sicher, dass alle Kombinationen der Bedingungszustände untersucht und ihnen bei Bedarf Aktionsfolgen zugeordnet werden.

Die systematische Regelentwicklung bei der Erstellung von Mehrtreffer -Entscheidungstabellen erfolgt dagegen aktions- und bedingungsorientiert. Daneben besteht die Möglichkeit, Entscheidungsregeln frei einzutragen.

2.2.4.1 Erstellung der Mehrtreffer-Entscheidungstabelle

Die aktionsorientierte Regelentwicklung eignet sich insbesondere zum Aufbau der Mehrtreffer-Entscheidungstabelle im Entscheidungsmatrix-Format. Dabei werden Bedingungen und Aktionen zunächst matrixförmig angeordnet. Die Reihenfolge der Bedingungen wird so gewählt, dass eine sequentielle Prüfung möglich ist. Die Reihenfolge der Aktionen wird so bestimmt, dass alle plausibel erscheinenden Aktionsfolgen von links nach rechts ablesbar sind.

Für jede Aktion wird nun geprüft, welche Voraussetzungen auf der Bedingungsseite gegeben sein müssen, damit diese Aktion ausgeführt werden kann. Oft ergeben sich bei dieser Prüfung mehrere Alternativen. Jede dieser Alternativen führt dann zu einer aktionsbezogenen Regel. Zum Abschluss wird eine aktionsbezogene Minimierung durchgeführt.

Die Entscheidungsmatrix hat eine klar definierte Untergrenze für die Regelanzahl: die Anzahl der Aktionen. Diese Untergrenze wird erreicht, wenn die Voraussetzung zur Ausführung jeder Aktion in Form einer einzigen Bedingungsanzeigerkombination formuliert werden kann.

Die aktionsorientierte Regelentwicklung stellt keinesfalls sicher, dass alle relevanten Bedingungskonstellationen abgebildet werden und eine Aktionsfolge zugeordnet erhalten. Umso mehr Bedeutung erhalten die Verifikationsprozeduren, deren Ausführung in Kombination mit der aktionsorientierten Regelentwicklung unerlässlich ist.

2.2.4.2 Bedingungsorientierte Regelentwicklung

Obwohl sich die aktionsorientierte Regelentwicklung auch bei der Erstellung einer Mehrtreffer-Entscheidungstabelle im Standardformat anwenden lässt, kann dabei häufig die bedingungsorientierte Regelentwicklung vorteilhafter eingesetzt werden. Nach entsprechender Anordnung von Bedingungen und Aktionen wird bei der ersten Bedingung der erste Bedingungsanzeiger in die Regelspalte eingetragen. Danach wird geprüft, ob bereits eine Auswirkung auf die Aktionen formuliert werden kann. Wenn eine oder mehrere Aktionen in ihrer Ausführung nur von dem untersuchten Zustand der ersten Bedingung abhängig sind, wird die Regelspalte durch Eintragen einer entsprechenden Aktionsanzeigerkombination komplettiert, und es wird der nächste Bedingungsanzeiger der ersten Bedingung in die zweite Regelspalte eingesetzt. Die anderen Bedingungen erhalten den Indifferenzanzeiger. Kann noch keine direkte Auswirkung auf die Aktionen festgestellt werden, dann können weitere Bedingungen mit berücksichtigt werden, wobei bei jeder Bedingung eine systematische Durchprüfung aller Bedingungsanzeiger empfohlen wird. Ist auf diese Weise eine Bedingungskonstellation gefunden worden, der eine Aktion oder Aktionsfolge zugeordnet werden kann, dann wird die Regel vervollständigt.

Durch Variation der Bedingungsanzeiger der in der zuletzt gefundenen Konstellation berücksichtigten sonstigen Bedingungen können möglicherweise weitere Regeln ermittelt werden. Der Übergang auf den nächsten Bedingungsanzeiger der ersten Bedingung erfolgt, wenn keine weiteren Regeln formuliert werden können.

Wenn alle Bedingungsanzeiger der ersten Bedingung durchgeprüft worden sind, werden alle anderen Bedingungen in gleicher Weise im Hinblick auf ihren Einfluss auf die Auslösung von Aktionen oder Aktionsfolgen überprüft. Vor der Formulierung jeder neuen Regel wird jedoch sichergestellt, dass nicht bereits eine identische Regel in der Tabelle vorhanden ist.

Nach gezielter Prüfung der Auswirkungen der letzten Bedingung wird die Entscheidungstabelle minimiert und verifiziert.

2.2.4.3 Freie Regeleintragung

Als großen Vorteil der Mehrtreffer-Entscheidungstabelle gegenüber der Eintreffer-Entscheidungstabelle könnte ein Anwender auch eine weitere, im Prinzip unsystematische Methode der Tabellenerstellung empfinden: die freie Regeleintragung.

Es wird nach Erstellung der Basisfolge von Bedingungen und Aktionen zugelassen, dass jede in sich plausible Entscheidungsregel ergänzt wird, wobei das Verhältnis jeder neu hinzukommenden Regel zu den bereits vorhandenen Regeln nicht untersucht wird. Nach Abschluss der Regeleintragung wird die Entscheidungstabelle insgesamt verifiziert und minimiert.

Diese Art der Tabellenerstellung bietet insbesondere dann Vorteile, wenn die Regeln aus unterschiedlichen Informationsquellen extrahiert werden müssen oder auch die Tabellenerstellung unter Zeitdruck–beispielsweise im Rahmen einer Besprechung oder eines Interviews mit Fachabteilungsangehörigen - erfolgt.

2.3 Pragmatik der Mehrtreffer-Entscheidungstabelle im Vergleich zur Eintreffer –Entscheidungstabelle

Während die Eintreffer-Entscheidungstabelle einer breiten Schicht von Anwendern bereits seit 1962 [1)] geläufig ist, werden ernstzunehmende Versuche mit der Mehrtreffer–Entscheidungstabelle erst seit wenigen Jahren unternommen. Im Vergleich zur Eintreffer–Entscheidungstabelle lassen sich sowohl Vor- als auch Nachteile der Mehrtreffertabelle aufzeigen.

2.3.1 Erstellungs- und Verifikationsaufwand

Während der Aufwand zur Definition von Bedingungen, Aktionen und Anzeigern bei beiden Tabellenarten identisch ist, verursacht die Regelentwicklung für die Mehrtreffer-Entscheidungstabelle im Allgemeinen deutlich weniger Aufwand als die Regelentwicklung für die Eintreffer–Entscheidungstabelle.

Das kann darauf zurückgeführt werden, dass alle Verfahren zur systematischen Regelentwicklung der Eintreffer-Entscheidungstabelle eine - zumindest implizite - Überprüfung jeder elementaren Bedingungsanzeigerkombina-

---

1) Im Jahre 1962 fand in New York das Decision Tables Symposium statt. Vgl. ACM (Decision Tables Symposium)

tion beinhalten, die elementare Bedingungsanzeigerkombination jedoch die zahlenmäßig am stärksten vertretene syntaktische Einheit der Entscheidungstabelle ist. Die Regelentwicklung für die Mehrtreffer–Entscheidungstabelle dagegen orientiert sich primär an Bedingungen und Aktionen oder lässt eine völlig freie Regeleintragung zu und erfordert deshalb weniger Entwicklungsschritte.

Fällt eine Bewertung der beiden Tabellenarten auf der Basis des Aufwands zur Regelentwicklung noch eindeutig zugunsten der Mehrtreffer -Entscheidungstabelle aus, so kehrt sich dieser Eindruck bei einer Bewertung des erforderlichen Verifikationsaufwands um. Die Verifikationsprozeduren für Mehrtreffer-Entscheidungstabellen sind nicht nur schwierig durchführbar, auch die Wahrscheinlichkeit, dass Fehler unentdeckt bleiben, ist größer. Zur Begründung sei auf die kaum vollständig durchführbare Vereinigung von Aktionsfolgen bei Regeln, deren Bedingungsanzeigerkombinationen nicht disjunkt sind, verwiesen.

Die Verifikation der Eintreffer-Entscheidungstabelle dagegen kann bei korrekter Anwendung der Tabellenerstellungsprozeduren keine Redundanzen und Unvollständigkeiten mehr aufdecken.

Sie hat eher den Charakter einer Kontrollprozedur, die weniger die fehlerfreie Abbildung des Entscheidungsfeldes als vielmehr die vollständige und fehlerfreie Anwendung der Tabellenerstellungsprozeduren überwacht.

Im gleichen Ausmaß, in dem sich die Notwendigkeit der Durchführung von Verifikationsprozeduren verringert, wächst jedoch die Attraktivität der Mehrtreffer-Entscheidungstabelle für den Benutzer. So weist beispielsweise eine bestimmte Klasse von Datenverarbeitungsproblemen eine starke logische Abhängigkeit zwischen Bedingungen und Aktionen derart auf, dass tendenziell jede Bedingung sich nur auf eine Aktion (oder wenige) auswirkt und jede Aktion auch nur von einer Bedingung (oder wenigen Bedingungen) beeinflusst wird. So konnte beispielsweise die Mehrtreffer-Entscheidungstabelle zur

Abbildung einer Plausibilitätsprüfung [1)]folgende (abstrakte)

Form aufweisen:

---

1) Auf die Anwendbarkeit der Mehrtreffer-Entscheidungstabelle zur Plausibilitätsprüfung weisen hin King; Johnson (Ambiguous decision tables) ,S. 289 f.

		1	2	3	4
B1	Datenelement 1 plausibel	N	-	-	-
B2	Datenelement 2 plausibel	-	N	-	-
B3	Datenelement 3 plausibel	-	-	N	-
B4	Datenelement 4 plausibel	-	-	-	N
A1	Fehlernachricht 1	X			
A2	Fehlernachricht 2		X		
A3	Fehlernachricht 3			X	
A4	Fehlernachricht 4				X

Bei dieser Mehrtreffer-Entscheidungstabelle mit ihrer charakteristischen diagonalen Anordnung von Bedingungs- und Aktionsanzeigern wird aufgrund der Transparenz ihrer Logik eine semantische Verifikation subjektiv als überflüssig empfunden.

Auch wenn in geringem Umfang *Bedingungskonstellationen* abgebildet werden müssen, da beispielsweise einige Unplausibilitäten nur durch gemeinsame Überprüfung von Datenelementen konstatiert werden können, bleibt oft die Abbildung so transparent, dass der Entscheidungstabellen -Anwender subjektiv keine Notwendigkeit einer Verifikation verspürt.[1)]

Ein weiteres Beispiel aus dem Bereich technischer Anwendungen sind Entscheidungstabellen zur Arbeitsplanung. Stehen für einen Arbeitsgang mehrere Maschinen zur Verfügung, dann erfolgt die Auswahl einer oder mehrerer geeigneter Maschinen über eine Mehrtreffer-Entscheidungstabelle, deren Regeln jeweils im Bedingungsanzeigerteil die relevanten technischen Charakteristika einer Maschine beschreiben und im Aktionsanzeigerteil lediglich die Auswahlanweisung definieren. Häufig sind die Bedingungen solcher Entscheidungstabellen mit überlappenden Bedingungsanzeigern versehen, da die exakte Beschreibung der technischen Merkmale jeder Maschine Vorrang vor der Forderung disjunkter Anzeiger im Hinblick auf eine einfachere Tabellenerstellung oder -verifikation hat. Diese Art der Entscheidungstabellen wird im allgemeinen Regel für Regel verifiziert, die logische Beziehung der Regeln zueinander dagegen ist vom Problem her irrelevant und bedarf keiner Verifikation.

---

1) Diese Beobachtung konnte der Verfasser vielfach in der Praxis machen.

Schließlich soll noch darauf hingewiesen werden, dass immer dann, wenn die erforderlichen Verifikationsprozeduren maschinell ausgeführt werden, das den Anwender veranlassen kann, die Mehrtreffer-Entscheidungstabelle der Eintreffer-Entscheidungstabelle vorzuziehen.

Allerdings verlangt das von ihm, dass die Entscheidungstabelle so erstellt wird, dass alle nur auf Syntaxbasis ausführbaren Verifikationsprozeduren auch semantische Relevanz besitzen.

2.3.2 Kommunikationseigenschaften [1)]

Zielsetzung der Übermittlung einer Entscheidungstabelle von einem personalen oder maschinellen Sender A zu einem personalen oder maschinellen Empfänger B kann entweder sein, dem Empfänger die gesamte abgebildete Entscheidungslogik bekanntzugeben, oder dem Empfänger das Treffen von Entscheidungen nach Maßgabe der Entscheidungstabelle zu ermöglichen.

Bei einer Kommunikation zwischen *Personen* kann im ersten Fall kein wesentlicher Unterschied in den Kommunikationseigenschaften der Mehrtreffer-Entscheidungstabelle im Vergleich zur Eintreffer–Entscheidungstabelle festgestellt werden. Beide Tabellenarten bilden eine vorgegebene Entscheidungslogik eindeutig und umkehrbar ab, so dass Sender und Empfänger über ein identisches Abbild des zu beschreibenden Entscheidungsfeldes verfügen. In vielen Fällen kann möglicherweise eine günstigere Kommunikationseigenschaft der Mehrtreffertabelle aufgrund ihrer geringeren Regel- und Anzeigeranzahl begründet werden.

Dagegen wird das Treffen von Entscheidungen nach Maßgabe einer Mehrtreffer-Entscheidungstabelle von einem personalen Empfänger als schwieriger empfunden werden als bei einer Eintreffertabelle. In jedem Ausführungsfall müssen *alle* Regeln durchgeprüft und in vielen Fällen auszuführende Aktionsfolgen erst durch Bilden einer Vereinigungsmenge ermittelt werden. Dies erhöht nicht nur den Aufwand zur Interpretation, sondern auch die Fehleranfälligkeit.

Bei Kommunikationsvorgängen mit *maschinellen* Empfängern lässt sich allerdings nur erhöhter Aufwand, nicht jedoch auch eine erhöhte Fehleranfälligkeit vermuten.

---

1) Bei der Erörterung der Kommunikationseigenschaften wird von den Auswirkungen der Primärsprache abstrahiert.

2.3.3 Konsequenzen für den praktischen Einsatz

In der Umgebung einer ausschließlich manuellen Anwendung werden Mehrtreffer-Entscheidungstabellen kaum eingesetzt werden können. Der erheblich höhere Verifikations- und Interpretationsaufwand wird nicht durch andere Vorteile kompensiert.

Eine rechnergestützte Anwendung von Mehrtreffer-Entscheidungstabellen kann jedoch durchaus sinnvoll sein. Unter der Voraussetzung einer weitgehenden Automatisierung von Verifikationsprozeduren können die Vorteile einer einfacheren Tabellenerstellung durchaus die bekannten Nachteile der Mehrtreffer-Entscheidungstabelle überwiegen. Auch in einer Umgebung rechnergestützter Anwendung der Entscheidungstabellentechnik muss jedoch der Eintreffer-Entscheidungstabelle in allen Fällen der Vorzug gegeben werden, in denen personale Aufgabenträger Entscheidungen auf der Basis van Entscheidungstabellen fällen müssen.

Die Gefahr von Interpretationsfehlern verbietet hier die Anwendung von Mehrtreffer-Entscheidungstabellen.

Teil III: Anwendungsmöglichkeiten der Entscheidungstabellentechnik bei der Gestaltung rechnergestützter Informationssysteme

3. Der Systemcharakter rechnergestützter Informationssysteme und die Phasen deren Gestaltung

Die Gesamtheit der Informationen einer irgendwie abgrenzbaren organisatorischen Einheit, deren logische Beziehungen zueinander, die diese Informationen verarbeitenden Subjekte und alle Informationsverarbeitungsprozesse als *System* zu betrachten, ist schon seit längerem in Theorie und Praxis Brauch. Die Elemente rechnergestützter Informationssysteme und die Art ihrer Beziehungen zueinander sollen deshalb in einem ersten Abschnitt kurz definiert werden.

Die *Gestaltung* eines rechnergestützten Informationssystems ist ein in mehrere Phasen unterteilbarer Prozess. Das im zweiten Abschnitt entworfene Modell eines Gestaltungsprozesses soll die Grundlage für eine in den nächsten Kapiteln vorgenommene Beurteilung der Einsatzmöglichkeiten von Entscheidungstabellen bilden.

3.1 Elemente eines rechnergestützten Informationssystems

Als erste Elementkategorie eines rechnergestützten Informationssystems sind die *Informationsverarbeitungsaufgaben* zu nennen. Sie lassen sich aus der Gesamtaufgabe ableiten, die von der organisatorischen Einheit erfüllt werden muss, deren Informationssystem analysiert wird. Als besonderes Kennzeichen rechnergestützter Informationssysteme gilt, dass eine Teilmenge dieser Informationsverarbeitungsaufgaben automatisch erfüllt wird.

Die Erfüllung von Informationsverarbeitungsaufgaben erfordert die Bereitstellung von Eingabeinformationen, und sie führt zur Entstehung von Ausgabeinformationen. Generell können *Informationen* als zweite Elementkategorie des Informationssystems angesehen werden.

Im rechnergestützten Informationssystem lässt sich eine Teilmenge dieser Informationen maschinell erfassen, speichern und transformieren.

---

1) Zu diesem Abschnitt vgl. Strunz; Bischoff; u.a. (Gestaltung von Informationssystemen), S. 11 f.

Die *Verarbeitungsprozesse*, die zur Erfüllung der Informationsverarbeitungsaufgaben ausgeführt werden müssen, bilden die dritte Elementkategorie. Ein Teil der Verarbeitungsprozesse muss zur automatischen Ausführung durch den Rechner geeignet sein.

Personale und maschinelle *Aktionsträger* bilden die vierte Elementkategorie eines rechnergestützten Informationssystems.

Schließlich können die zwischen den Aktionsträgern bestehenden *Kommunikationswege* als fünfte und letzte Elementkategorie bezeichnet werden.

Vielfältige Beziehungen bestehen nicht nur zwischen den Elementen einer Elementkategorie, sondern auch zwischen Elementen verschiedener Kategorien.

Die Komplexität eines rechnergestützten Informationssystems wird sowohl durch quantitative als auch durch qualitative Merkmale bestimmt. So erhöht sich der Komplexitätsgrad bei zunehmender Anzahl von Elementen in jeder Kategorie, aber auch bei einer Intensivierung der Beziehungen der Elemente einer Kategorie oder verschiedener Kategorien zueinander.

3.2 Ein Phasenmodell der Systemgestaltung

Der Lebenszyklus eines Informationssystems lässt sich zunächst grob in die drei Phasen

Systemplanung,
Systemimplementierung und
Systemanwendung

unterteilen. [1)]

Teilaufgaben der *Systemgestaltung* lassen sich in allen Phasen des Lebenszyklus nachweisen.

Im Rahmen der Systemplanung lässt sich ein vorwiegend analytischer Teil - die Systemanalyse - und ein vorwiegend synthetischer Teil - der Systementwurf - unterscheiden. Die Systemanalyse wiederum kann in eine Phase der Projektinitialisierung und eine Phase der Istaufnahme unterteilt werden.

---

1) Strunz; Bischoff; u.a. (Gestaltung von Informationssystemen), S. 13

Um ein Systemgestaltungsprojekt zu initialisieren, ist es zunächst erforderlich, ein Datenverarbeitungsproblem zu erkennen und zu definieren.[1)] Eine darauf folgende Analysephase soll Aufschluss über die Durchführbarkeit einer Systementwicklung geben, die zum einen das erkannte Problem löst, zum anderen gewisse als notwendig erachtete Nebenbedingungen, wie beispielsweise die Wirtschaftlichkeit der Aufgabenerfüllung, sicherstellt. Im Rahmen dieser Phase wird eine Vorstudie [2)] erstellt, auf deren Grundlage die Entscheidung über die Fortführung oder Einstellung des Projekts getroffen werden kann.

Bei positiver Entscheidung folgt der Projektinitialisierung die Istaufnahme [3)]. Aufgenommen und analysiert werden zunächst Strukturelemente der untersuchten organisatorischen Einheit, wie beispielsweise ihre Aufgaben, ihre Organisationsstruktur, ihre personelle Struktur und ihre Kosten. Eine Erfassung und Analyse ihrer Informationsverarbeitungsprozesse folgt. Ergänzend werden Untersuchungen zum Informationsbedarf durchgeführt.

Der *Systementwurf* als zweite Phase der Systemplanung lässt sich in einen organisatorischen und einen systemtechnischen Entwurfsteil untergliedern.

Im Rahmen der organisatorischen Komponente werden alle Elementkategorien des Informationssystems bis zu einem Detaillierungsgrad entworfen, der noch die Abstraktion von spezifischen Auswirkungen der Rechneranwendung erlaubt.

Nach einer erst in den letzten Jahren entwickelten Entwurfstechnik werden zunächst die Informationsverarbeitungsaufgaben des neuen Systems mit ihren Beziehungen zueinander bestimmt. Als Beziehung zwischen zwei Informationsaufgaben gilt der Informationsfluss. Die als Eingabe zur Aufgabenerfüllung benötigten Informationen und die durch die Aufgabenerfüllung entstehenden Informationen werden zusammen mit jeder Aufgabe definiert. Erleichtert wird die Definition von Aufgaben, Beziehungen und Informationen durch die Benutzung von Modellen, die bereits die wichtigsten Informationsverarbeitungsaufgaben des untersuchten Problems in ihrem integrativen Zusammenhang sowie die benötigten bzw.

---

1) Vgl. dazu Schmitz; Seibt (Anwendungsorientierte Informatik), S. 144 f.; und Bingham; Davies (Systems analysis), S. 8 ff.

2) Zum Inhalt einer solchen Vorstudie vgl. Bingham; Davies (Systems analysis), S. 11 ff.; Chandor; Graham; Williamson (Systems analysis), S. 37 ff.; Clifton (Systems analysis), S. 22

3) Zum Definitionsbereich des Begriffs "Istaufnahme" vgl. Pärli (Istaufnahme), S. 29 f.

erzeugten Informationen beschreiben.[1)] Aufgabengruppen mit starker integrativer Verflechtung können zu Subsystemen zusammengefasst werden.

Auch die Art und Weise der *Aufgabenerfüllung* zu beschreiben, ist Bestandteil der organisatorischen Entwurfsphase. Hier besteht jedoch nur die Notwendigkeit, die *Methode* der Aufgabenerfüllung zu beschreiben und nicht etwa einen datenverarbeitungstechnisch günstigen Algorithmus zu entwickeln. Durchaus sinnvoll ist es jedoch, bereits im Rahmen des organisatorischen Entwurfs die Aufgaben des Informationssystems in manuell auszuführende und zu automatisierende zu klassifizieren.

Der Entwurf der Elementkategorie "Aktionsträger" ergibt einen Stellenplan für das zum Betrieb des Informationssystems benötigte Personal in Datenverarbeitungsabteilung und Fachabteilungen sowie detaillierte Beschreibungen jeder Stelle mit Verantwortungs- und Kompetenzbereich.

Auch der Entwurf der formalen Kommunikationswege zählt zum Aufgabenspektrum des organisatorischen Entwurfs. Der organisatorischen Gestaltung liegen die Informationserfordernisse der Aufgaben, die logische Beziehung der Aufgaben und die Zuordnung der Aufgaben auf die Aktionsträger zugrunde. Die Notwendigkeit des Austauschs von Informationen zwischen zwei Aktionsträgern begründet das Erfordernis eines formalen Kommunikationsweges.

Der auf den organisatorischen Entwurf folgende systemtechnische Entwurf beinhaltet eine Konkretisierung der Ergebnisse des organisatorischen Entwurfs unter Berücksichtigung des Rechners als Aktionsträger.

Die Informationsverarbeitungsaufgaben werden stärker detailliert und zu einer datenverarbeitungstechnisch günstigen Ablaufstruktur vereinigt. Zur Detaillierung der Aufgaben eignet sich besonders die Methode der funktionalen Modularisierung, die zu einer hierarchischen

---

1) Als bekannteste Modelle gelten das Kölner Integrationsmodell (KIM) und das AEG-Telefunken-Modell. Vgl. dazu Grochla (Konzept des Kölner Integrationsmodells), S. 34 ff.; Grochla; Garbe; u.a. (Kölner Integrationsmodell), S. 189 ff.; Köster; Hetzel (Datenverarbeitung mit System), S. 13 ff.

Aufgliederung einer Aufgabe führt.[1)]

Die zu den definierten Aufgaben ermittelten Eingabe -/Ausgabeinformationen werden zu Dateien zusammengefasst, wobei logische Strukturen der Informationen zu entsprechenden Datenstrukturen führen.[2)] In der Regel werden auch die Speichermedien bestimmt. Die im Rahmen des organisatorischen Entwurfs nur grob beschriebenen Methoden der Aufgabenerfüllung werden detailliert sowie geeignete Algorithmen entwickelt[3)] oder ausgewählt. Alle Verarbeitungsprozesse müssen nach Abschluss dieser Teilphase hinreichend ausführlich spezifiziert sein. Bei konsequenter Anwendung modularer Entwurfstechniken müssen außerdem die Makrostruktur des modularen Programmsystems als Definition der Programmmoduln und ihrer logischen Beziehungen sowie der Inhalt jedes Moduls definiert werden.

Im Rahmen des Aktionsträgerentwurfs werden Hardware -/Systemsoftware -Erfordernisse des neuen Systems definiert.

Auch die Kommunikationswege werden besonders an den Schnittstellen zwischen menschlichen und maschinellen Aktionsträgern intensiv analysiert. Die Wirksamkeit eines rechnergestützten Informationssystems kann entscheidend durch die Art des Informationsaustauschs zwischen Mensch und Maschine beeinflusst werden. Zu entwerfen sind außerdem die Datenübertragungswege innerhalb der ADV-Anlage zwischen Zentraleinheit und Ein-/Ausgabegeraten sowie peripheren Speichern.

Nach Abschluss der Systemplanung beginnt die *Systemimplementierung*, die einen systemtechnischen und einen organisatorischen Teil umfasst. Auf der Basis der als Ergebnis des systemtechnischen Entwurfs erstellten Spezifikationen werden Programme oder Programmmoduln codiert und getestet.[4)] Programme oder Programmmoduln müssen außerdem auf ihre

---

1) Die hierarchische Aufgliederung von Aufgaben wird als Entwurfsmethode beispielsweise im Rahmen des Systems HIPO empfohlen. Vgl. IBM (Problemlösung), S. 310/4 ff.
Bei der derzeit sehr "modernen" Aufgabenanalyse durch hierarchische Aufgliederung handelt es sich offensichtlich um nichts anderes als um das in der Allgemeinen Organisationslehre schon seit längerem bekannte Prinzip der Aufgabenanalyse nach dem Merkmal "Verrichtung". Vgl. Kosiol (Organisation), S. 49 f.

2) Vgl. Hartmann; Matthes; Proeme (Systems handbook), Abschnitt 3.4, S. 4 ff.

3) Zum Algorithmenentwurf vgl. Wedekind (Systemanalyse), S. 200 ff.

4) Zur systemtechnischen Implementierung vgl. Schmitz; Seibt (Anwendungsorientierte Informatik), S. 163 ff.

Funktionsfähigkeit im integrierten Ablauf überprüft werden. Die entworfenen Dateien werden eingerichtet und initialisiert. Wenn erforderlich, werden Zugriffsroutinen erstellt.

Die organisatorische Implementierung umfasst alle Maßnahmen zur Vorbereitung der Integration des neuen Systems in seine natürliche Umgebung. Alle Systemdateien werden mit aktuellen Daten gefüllt, die Systembenutzer und das Bedienungspersonal werden geschult, und es erfolgt schließlich eine Umstellung vom alten auf das neue System.[1)]

Die *Systemanwendung* beschließt den Lebenszyklus eines Informationssystems. Im Normalfall umfasst die Systemanwendung eine größere Zeitspanne als Systemplanung und Systemimplementierung zusammen. Neben allen Aufgaben, die den laufenden Betrieb des Systems betreffen, sowie Kontrollen, die fehlerhafte Reaktionen des Systems erkennen sollen, enthält auch die Systemanwendung gestalterische Aufgaben. Im Rahmen der Systemwartung nämlich werden laufend Anpassungen des Systems an veränderte Anforderungen vorgenommen, die ihrerseits wiederum geplant und implementiert werden müssen.

---

1) Zur organisatorischen Implementierung vgl. die ausführlicheren Darstellungen bei Schmitz; Seibt (Anwendungsorientierte Informatik), S. 176 ff.

4. Entscheidungstabellen als Hilfsmittel der Systemplanung

In der Phase der Systemplanung lassen sich Entscheidungstabellen überall dort anwenden, wo es gilt, Verarbeitungsprozesse detailliert zu beschreiben. Das ist sowohl bei der Systemanalyse als auch im Rahmen des Systementwurfs der Fall. Der Diskussion der Anwendungsmöglichkeiten der Entscheidungstabellentechnik in diesen Bereichen soll eine Diskussion der Grundlagen einer rechnergestützten Anwendung der Entscheidungstabellentechnik in der Systemplanungsphase folgen.

4.1 Einsatzmöglichkeiten der Entscheidungstabellentechnik bei der Systemanalyse

Als Teilphasen der Systemanalyse wurden die Projektinitialisierung mit Problemerkennung und Durchführbarkeitsanalyse sowie die Istaufnahme genannt.

4.1.1 Entscheidungstabellen in der Phase der Projektinitialisierung

Das zu gestaltende System wird in der Phase der Projektinitialisierung noch nicht so detailliert beschrieben werden, dass die Anwendung von Entscheidungstabellen notwendig erscheint. Wenn im Rahmen der Vorstudie auf die Leistungen des vorgeschlagenen Systems eingegangen wird, ist es eher üblich, Art und Qualität der durch das neue System bereitgestellten Informationen, nicht jedoch die dazu benötigten Verarbeitungsprozesse detailliert zu konzipieren.

Während der Phase der Projektinitialisierung sind jedoch Entscheidungen darüber erforderlich, welche der erkannten Datenverarbeitungsprobleme in Form einer Vorstudie weiterverfolgt werden sollen. Wo diese Entscheidung nicht aufgrund weniger Kriterien [1)] getroffen werden kann, wie beispielsweise

- offensichtliche Wirtschaftlichkeitsvorteile,
- Wunsch des Managements,
- technische Durchführbarkeit,
- vorhandene Entwicklungskapazitäten bei der Datenverarbeitungsabteilung,
- Notwendigkeit der Vervollständigung eines integrierten Systems,
- dringendes Erfordernis der Unternehmung, das keiner sonstigen Rechtfertigung bedarf,

---

1) Vgl. Bingham; Davies (Systems analysis), S. 8 ff.

kann aufgrund einer erheblich höheren Anzahl von Kriterien und bestehender Interdependenzen der Auswahlvorgang so schwierig werden, dass der Wunsch nach einem formalisierten Verfahren besteht. Hier bietet die Entscheidungstabellentechnik die Möglichkeit, den Auswahlvorgang so exakt zu beschreiben, dass das Auswahlergebnis nachvollziehbar und damit nachprüfbar wird. Zusätzliche Entscheidungskriterien können in Form zusätzlicher Entscheidungstabellen berücksichtigt werden. Das gesamte Auswahlverfahren kann so bei Bedarf objektiviert werden, was sich bei einem starken Missverhältnis zwischen Projektwünschen der Fachhabteilungen einerseits und Entwicklungskapazität der Datenverarbeitungsabteilung andererseits als recht günstig erweisen kann.

4.1.2 Istaufnahme mit Entscheidungstabellen

Bei der Istaufnahme von Datenverarbeitungsprozessen entfällt ein erheblicher Teil des Erhebungs- und Analyseaufwands auf die Untersuchung der Verarbeitungsprozeduren. Die Anwendung der Entscheidungsstabellentechnik als Beschreibungs- und Analysehilfsmittel empfiehlt sich aus mehreren Gründen.

Zum einen erweist sie sich in der Hand des Systemplaners als ein ausgezeichnetes Analysehilfsmittel. Die geringen Kenntnisse des Untersuchungsgebietes, über die ein Systemplaner bei Projektbeginn verfügt, erlauben es ihm nicht, alle Sonderfälle der Verarbeitung logisch abzuleiten. Da sich diese Sonderfälle auch nur selten aus der Analyse vorhandener Unterlagen, aus Berichten oder Interviews vollständig erkennen lassen, sind Erfassungslücken unvermeidbar, die wiederum einen lückenhaften Systementwurf nach sich ziehen können. Bei Anwendung von Entscheidungstabellen als Analyse- und Beschreibungsmittel wird dieses Problem entschärft. Aus einer strukturellen Betrachtung heraus verliert der Unterschied von Normalfällen und Sonderfällen der Bearbeitung seine Bedeutung, vielmehr gibt es nur unterschiedliche Verarbeitungsalternativen. Sofern zu den ermittelten Bedingungen nicht alle Konstellationen mit Aktionen versehen worden sind, gelingt es dem Systemplaner, die Lücken mit Hilfe der Verifikationsmethoden der Entscheidungstabellentechnik klar zu lokalisieren.[1)] Er wird dadurch in eine günstige Position gebracht: Nunmehr kann er sich gezielt nach

---

1) Einen Hinweis auf die Vollständigkeitskontrolle in der Istaufnahme mit Hilfe von Entscheidungstabellen gibt auch Pärli (Istaufnahme) S. 143

den fehlenden Angaben erkundigen. Auch logische Widersprüche in den möglicherweise aus unterschiedlichen Informationsquellen stammenden Angaben zu einem Informationsverarbeitungsprozess deckt er mit Hilfe der Entscheidungstabellenverifikation leicht auf.

Die Entscheidungstabelle in der Istaufnahme erweist sich zum anderen als leistungsfähiges Kommunikationsmittel. Organisatoren, Systemplaner und Fachabteilungsangehörige arbeiten gemeinsam daran, die wesentlichen Merkmale des bestehenden Systems dem Untersuchungsziel entsprechend zu erfassen. Bei der Darstellung komplexer Entscheidungslogik erweist sich die Entscheidungstabelle als Hilfsmittel, das von allen an der Istaufnahme Beteiligten akzeptiert werden kann.

Die Entscheidungstabelle kann weder als problembezogenes Beschreibungsmittel noch als reines Werkzeug der Datenverarbeitung bezeichnet werden: Sie ermöglicht die Erfassung und Analyse von Entscheidungsprozessen auf logischer und damit allgemeinverständlicher Grundlage.

Zu beachten bleibt jedoch, dass sich die positiven Auswirkungen der Entscheidungstabelle als Kommunikationsmittel auf die Beschreibung struktureller Aspekte beschränken. Die in der Istaufnahme benutzte Primärsprache zur Beschreibung von Bedingungen, Aktionen und Anzeigern bestimmt die Kommunikationsfähigkeit der Istaufnahmeergebnisse in erheblichem Ausmaß mit.

4.2 Einsatzmöglichkeiten der Entscheidungstabellentechnik beim Systementwurf

In der Phase des Systementwurfs lassen sich Entscheidungstabellen immer dann anwenden, wenn Datenverarbeitungsprozesse detailliert beschrieben werden müssen. Das ist sowohl im Rahmen der organisatorischen als auch der systemtechnischen Komponente des Systementwurfs der Fall.

4.2.1 Entscheidungstabelleneinsatz im Rahmen organisatorischer Entwurfsarbeiten

Im Rahmen organisatorischer Entwurfsarbeiten muss vielfach zusätzlich zur Definition einer Aufgabe auch die Methode der Aufgabenerfüllung spezifiziert werden, da diese während der Implementierungsphase nicht aus der Zuordnung von eingehenden Informationen zu ausgehenden Informationen erschlossen werden kann. Zur Darstellung der Aufgabener-

füllungsmethoden sind verschiedene sprachliche Ausdrucksmittel in Gebrauch, von verbalen Beschreibungen in natürlicher Sprache oder einer daraus abgeleiteten Fachsprache über formale Sprachen unterschiedlichster Art bis zu symbolischen Darstellungen.

Die Anwendung von Entscheidungstabellen zur Beschreibung einer Aufgabenerfüllungsmethode empfiehlt sich insgesamt oder partiell immer dann, wenn die Aufgabenerfüllung Entscheidungsvorgänge einschließt. Wenn die zur Beschreibung von Bedingungen. Aktionen und Anzeigern der Entscheidungstabellen verwendete Sprache dem nicht entgegensteht, erhöht die Anwendung von Entscheidungstabellen die Transparenz des Entwurfs beträchtlich. Die zur Aufgabenerfüllung benötigten Informationen werden in der Regel im Bedingungsteil der Entscheidungstabelle auf ihren Inhalt überprüft. Der spätere Implementator des Informationsverarbeitungsprozesses kann dann durch Zusammenstellung aller Bedingungen der in der Beschreibung aufgeführten Entscheidungstabellen leicht einen Überblick über die wesentlichen Eingangsinformationen des Aufgabenerfüllungsprozesses erhalten.

Die Art der Ausgangsinformationen der Aufgaben lässt sich aus dem Aktionsteil der in der Beschreibung enthaltenen Entscheidungstabellen erschließen. Der spätere Implementator kann leicht erkennen. welche Informationsarten durch die beschriebenen Aktionen beeinflusst werden. Durch Studium der Entscheidungsregeln lässt sich dann jeder Transformationsvorgang im Detail nachvollziehen.

Der Systemplaner kann bei Anwendung von Entscheidungstabellen nicht nur gute Kommunikationseigenschaften seines Entwurfs erwarten. Die ausgereiften Verifikationsprozeduren der Entscheidungstabellentechnik erlauben ihm die Sicherstellung von logischer Konsistenz und Vollständigkeit seines Entwurfs in einem Ausmaß. das kein anderes in der Systemplanung gebräuchliches sprachliches Ausdrucksmittel bietet.

Zu berücksichtigen bleibt jedoch, dass die positiven Auswirkungen der Entscheidungstabellenanwendung auf strukturelle Aspekte des Aufgabenerfüllungsprozesses beschränkt bleibt. Zur Feststellung der Konstellation von Eingangsinformationen ungeeignete Bedingungsprüfungen können über Verifikationsprozeduren der Entscheidungstabellentechnik ebenso wenig erkannt werden wie zur Erzeugung von Ausgabeinformationen ungeeignete Aktionen. Hier wird deutlich, dass die Entscheidungstabellentechnik nur eines von mehreren methodischen Hilfsmitteln ist, deren sich der Systemplaner in der Entwurfsphase bedienen muss.

4.2.2 Systemtechnischer Entwurf mit Entscheidungstabellen

Während in der Phase des organisatorischen Entwurfs die Spezifikation von Informationsverarbeitungsprozessen vorwiegend unter der Zielsetzung erfolgt, die Methode der Aufgabenerfüllung transparent zu machen, erfolgt in der Phase des systemtechnischen Entwurfs eine Detaillierung derjenigen Informationsverarbeitungsprozesse, die automatisiert werden sollen, im Hinblick auf eine optimale maschinelle Ausführung: Sie werden algorithmisiert. Der Algorithmus integriert die problembezogenen Verarbeitungserfordernisse einerseits mit den rechnerbezogenen Verarbeitungsmöglichkeiten andererseits.

Als sprachliches Ausdrucksmittel zur Beschreibung von Algorithmen wird überwiegend der Programmablaufplan benutzt. Mit Hilfe untereinander verknüpfter Sinnbilder[1)] zeigt der Programmablaufplan die Gesamtheit aller beim Programmablauf möglichen Wege auf. Als Programmablauf wird die zeitliche Beziehung zwischen den Teilvorgängen verstanden, aus denen sich die folgerichtige Ausführung eines Programms zusammensetzt.[2)]

Programmablaufpläne können unterschiedliche Detaillierungsgrade aufweisen. In ihrer detailliertesten Form erlauben sie eine l:l -Codierung in der Implementierungsphase. Jedes Sinnbild des Programmablaufplans führt dann genau zu einer Anweisung der benutzten Programmiersprache. Programmablaufpläne, die während des systemtechnischen Entwurfs entstehen, sind in der Regel sehr viel abstrakter ausgelegt. Auf eine Detaillierung wird immer dann verzichtet, wenn entsprechendes Detailwissen beim Programmierer vorausgesetzt werden kann. Der Systemplaner legt seinen Schwerpunkt auf die Erläuterung problembezogener Verarbeitungsvorgänge und verliert sich nicht in datenverarbeitungstechnischen Details, wie beispielsweise der Beschreibung der Lesesteuerung und Satzauswahl sowie der Gruppenverarbeitung bei kommerziellen Datenverarbeitungsproblemen.

Programmablaufpläne sind formal durchaus mit Entscheidungstabellen vergleichbar. Die Sinnbilder und die Regeln zur Bildung von Ablaufstrukturen können ebenso wie die Entscheidungstabellen als *Strukturspra*che verstanden werden. Programmablaufpläne haben ebenso wie Entscheidungsstabellen den Charakter einer Sekundärsprache, die der Ergänzung durch

---

1) Diese Sinnbilder sind genormt nach DIN 66001. Vgl. DNA (Normen), S. 93 ff.
2) Vgl. Schmitz; Seibt (Anwendungsorientierte Informatik), S. 163 ff.

Ausdrücke einer Primärsprache bedürfen, damit die Semantik des beschriebenen Verarbeitungsprozesses interpretierbar und nachvollziehbar ist. Diese Primärsprache kann ebenso wie bei der Entscheidungstabelle eine natürliche Sprache oder eine formale Sprache sein.

Die Verwendung von Entscheidungstabellen im Rahmen des systemtechnischen Entwurfs stellt eine Alternative zur Anwendung von Programmablaufplänen dar. Ihr Einsatz wäre dann sinnvoll, wenn sie ebenso wie ein Programmablaufplan die Algorithmisierung eines Aufgabenerfüllungsprozesses gestatten, darüber hinaus jedoch pragmatische Vorteile gegenüber Programmablaufplänen aufweisen.

Ein Algorithmus ermöglicht die Umformung gegebener Eingabeinformationen aufgrund eines Systems von Transformationsregeln in Ausgabeinformationen. Es müssen sechs grundlegende Bedingungen erfüllt sein, damit ein Algorithmus formuliert werden kann.[1)]

1. Das System G der Größen, die ineinander umgeformt werden sollen, muss definiert sein.
2. Es wird vorausgesetzt, dass die Umformung der Größen in Arbeitsschritten vor sich geht. Jeder Arbeitssehritt besteht in der Anwendung einer Regel.
3. Die Beschreibung muss vollständig sein.
4. Die Beschreibung muss endlich sein.
5. Das System zulässiger Operationen muss angegeben werden.
6. Die Sprache, in der die Regeln formuliert werden, muss angegeben werden.

Die erste dieser Grundbedingungen wird durch die Verwendung von Programmablaufplänen oder Entscheidungstabellen nicht berührt.

Die zweite der Grundbedingungen erfordert offensichtlich eine Besehreibungsform, mit der die sequentielle Aufeinanderfolge der Arbeitssehritte darstellbar ist. Sehr konsequent entspricht der Programmablaufplan dieser Forderung. Sequentiell aufeinander folgende Arbeitsschritte werden durch sequentiell hintereinander angeordnete Sinnbilder beschrieben. Wo dies aus darstellungstechnischen Gründen nicht möglich ist, sorgen Pfeile oder Konnektoren für die richtige Interpretation.

---

1) Vgl. zum folgenden Gellert; Küstner u.a. (Mathematik), S. 780 ff.

Auch Entscheidungstabellen erlauben eine sequentielle Darstellung von Arbeitsschritten. Die Aufeinanderfolge der Entscheidungstabellen innerhalb eines Entscheidungstabellenverbunds ist durch die Verküpfungsanweisungen klar definiert. Die Abarbeitung jeder Tabelle erfolgt in der Weise, dass zuerst die Bedingungsprüfungen erfolgen und danach die Aktionen ausgeführt werden. Die Reihenfolge, in der Bedingungen geprüft und Aktionen ausgeführt werden, ist durch die Anordnung von Bedingungen und Aktionen in ihrer jeweiligen Basisfolge klar definiert. Die Entscheidungstabelle verlässt jedoch ganz konsequent die sequentielle Darstellungsform, wenn das Problem keine sequentiellen Arbeitsschritte erfordert. Alternative Arbeitsschrittfolgen werden in Form parallel angeordneter Entscheidungsregeln dokumentiert.

In Programmablaufplänen sind alternative Arbeitsschrittfolgen dagegen nur schwer erkennbar. Oft lassen sie sich nur dann aufdecken, wenn der Programmablauf mit Hilfe von Testdaten simuliert wird.

Programmablaufpläne weisen gegenüber Entscheidungstabellen jedoch den Vorteil auf, auch sequentielle Abläufe ohne Entscheidungslogik beschreiben zu können. Die Entscheidungstabellensprache lässt sich jedoch leicht um die bedingungslose Entscheidungstabelle, auch "Aktionstabelle" [1)] genannt, erweitern. Die Aktionstabelle besteht nur aus einer Aktionsfolge, die ohne Prüfung einer Bedingung bei jedem Tabellendurchlauf vollständig ausgeführt wird.

Abschließend kann konstatiert werden, dass sowohl der Programmablaufplan als auch die Entscheidungstabelle die zum Aufbau eines Algorithmus erforderliche Definition von Arbeitsschritten gestatten. Die Entscheidungstabelle trennt jedoch deutlicher sequentielle Arbeitsschritte von alternativen Arbeitsschritten und kann deshalb aus pragmatischer Sicht zur Beschreibung von Algorithmen, die einen nennenswerten Anteil an Entscheidungslogik beinhalten, als vorteilhafter eingestuft werden.

Die *dritte* Grundbedingung, die eine vollständige Beschreibung fordert, kann offensichtlich bei Anwendung von Entscheidungstabellen einfacher als bei Programmablaufplänen sichergestellt werden. Der Programmablaufplan erlaubt nur eine punktuelle strukturorientierte Vollständigkeitsprüfung. Jedes im Programmablaufplan enthaltene Bedingungssymbol kann zum Anlass genommen werden, die Vollständigkeit des Programmab-

---

1) In den Ausführungen zur Theorie der Entscheidungstabelle wurde darauf verzichtet, diese Entscheidungstabellenart darzustellen, da sie aus theoretischer Sicht nicht erklärungsbedürftig erscheint.

laufs bei allen Zuständen der *Bedingung* zu prüfen. Die Entscheidungsstabelle dagegen ermöglicht die strukturelle Vollständigkeitsprüfung von *Bedingungskonstellationen* und erlaubt damit eine wesentlich intensivere strukturorientierte Vollständigkeitsprüfung des Algorithmus.

Zur Erfüllung der *vierten* und *fünften* Grundbedingung leisten Programmablaufplan oder Entscheidungstabelle keinen Beitrag.

Die *sechste* Grundbedingung fordert, dass die Sprache, in der die Umformungsregeln formuliert werden, angegeben sein muss. Bei einer Gegenüberstellung von Programmablaufplan und Entscheidungstabelle als Sprachen zur Beschreibung der Struktur eines Algorithmus lässt sich der größere Umfang klar definierter Bildungs- und Transformationsregeln der Entscheidungstabelle nicht übersehen. Sofern die Qualität der Sprache mit berücksichtigt wird, müssen der Entscheidungstabelle pragmatische Vorteile zuerkannt werden.

Insgesamt überwiegen die pragmatischen Vorteile der Entscheidungstabelle diejenigen des Programmablaufplans. Diese Vorteile überwiegen umso mehr, je größer der Anteil der Entscheidungslogik innerhalb eines Algorithmus ist.

4.3 Grundlagen einer rechnergestützten Entwicklung von Entscheidungstabellen

Die automatisierte Verarbeitung der Ausdrücke einer formalen Sprache beschränkt sich auf die syntaktische Dimension dieser Sprache. Einen Rechner zur Entwicklung von Entscheidungstabellen zu benutzen, bedeutet, nur bei syntaktischen Bildungs- und Transformationsoperationen im Rahmen der Entwicklung von Entscheidungstabellen die Unterstützung eines Rechners zu genießen. Je mehr es gelingt, semantische Bildungs - und Transformationsoperationen auf eine syntaktische Dimension zu reduzieren, desto umfassender ist die Rechnerunterstützung. Im nachfolgenden ersten Unterabschnitt soll die Bedeutung syntaktischer Verfahren im Rahmen einer formalen Betrachtung der Teilaufgaben bei der Erstellung von Entscheidungstabellen herausgearbeitet werden.

Da die rechnergestützte Erstellung von Entscheidungstabellen noch in den Anfängen ihrer Entwicklung steckt, kann versucht werden, die bereits vorliegenden Ergebnisse einer automatischen Behandlung syntaktischer Ausdrücke einer anderen und offensichtlich methodisch erheblich weiter entwickelten Strukturspra che analog in der Entscheidungsstabellentechnik anzuwenden: der Graphentheorie. Die *Graphentheorie*

wird anwendbar, wenn es gelingt, die Entscheidungstabelle und den Entscheidungstabellenverbund als Graph zu definieren. Im zweiten Unterabschnitt wird ein Versuch in dieser Richtung unternommen.

4.3.1 Teilaufgaben der Entwicklung von Entscheidungstabellen

Die Abgrenzung des Entscheidungsfeldes, das mit Hilfe von Entscheidungstabellen beschrieben werden soll, ist der erste Schritt in der Entwicklung von Entscheidungstabellen. Eine genauere Analyse des Entscheidungsfeldes muss dann Aufschluss über die erforderlichen Entscheidungsprozesse geben. Jeder Entscheidungsprozeß verläuft - formal betrachtet - in folgenden Phasen:

1. Variablen, die den Zustand des Informationssystems an entscheidungsrelevanten Punkten beschreiben, werden geprüft.
2. In Abhängigkeit vom Inhalt dieser Variablen wird ein geeigneter Algorithmus ausgewählt, der zur zielgerichteten Veränderung von Zustandsvariablen des Informationssystems geeignet ist.
3. Die Ausführung des ausgewählten Algorithmus verändert diese Zustandsvariablen.

Dieser Prozess lässt sich in Anlehnung an ein Schaubild von Langefors[1)] wie folgt veranschaulichen:

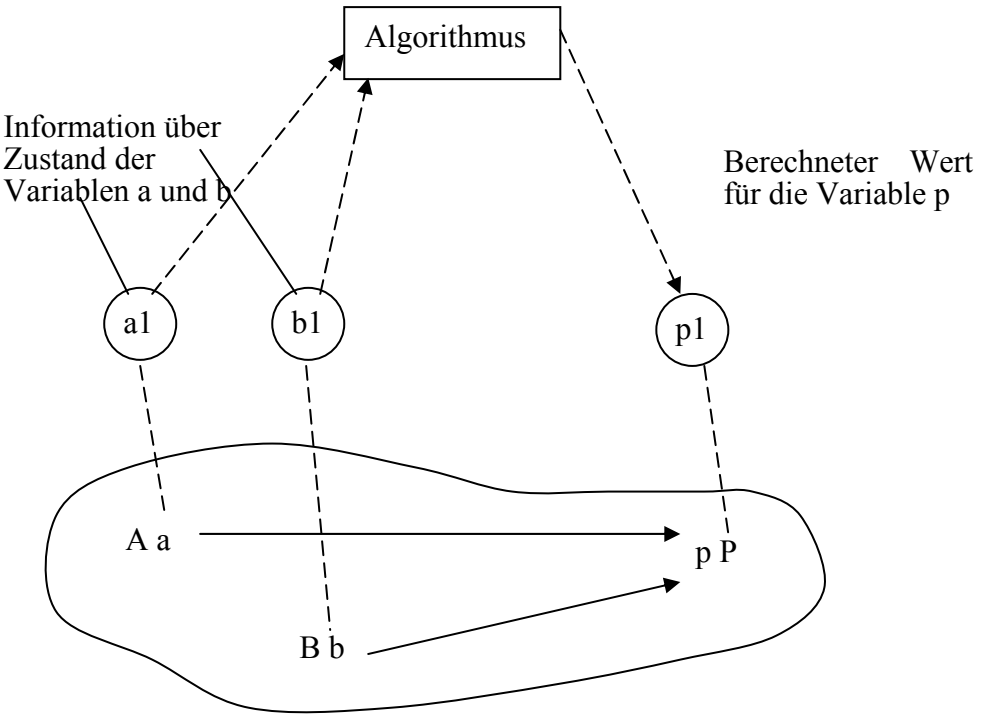


Abb. 5: Formale Phasen des Entscheidungsprozesses

1) Langefors (Information systems), S. 281

An den Punkten A, B und P eines Informationssystems lassen sich die Zustandsvariablen a, b und p messen. p ist von den Variablen a und b abhängig. Ein Algorithmus, der p berechnen soll, benötigt als Input den Zustand der Variablen a und b. Beispielsweise errechnet er p1 aus al und b1.

Der "Algorithmus" des Schaubildes kann in Form einer einzigen Entscheidungstabelle oder durch einen Entscheidungstabellenverbund realisiert sein. Bevor Entscheidungstabellen aufgebaut werden können, ist es demnach erforderlich, die wesentlichen Zustandsvariablen eines Informationssystems zu erfassen und ihre logischen Beziehungen zu analysieren. Insbesondere sind die Präzedenzrelationen zu ermitteln, die zwischen den Variablen bestehen, um zunächst grob in zu prüfende Variablen und zu beeinflussende Variablen unterscheiden zu können. Für Variablen, deren Zustand geprüft werden soll, müssen danach entsprechende Prüfprozeduren definiert werden. Veränderungsprozeduren schließlich werden benötigt, um die zu beeinflussenden Variablen zielgerichtet umformen zu können. Erst zu diesem Zeitpunkt wird es dann möglich, die Entscheidungslogik detailliert zu formulieren, das heißt, unterschiedliche Veränderungsprozeduren verschiedenen Zustandskonstellationen der zu prüfenden Variablen zuzuordnen.

Die besondere Problematik, die es bei einer rechnergestützten Erstellung von Entscheidungstabellen zu bewältigen gilt, liegt offensichtlich in der Ermittlung der entscheidungsrelevanten Zustandsvariablen des Informationssystems und ihrer logischen Beziehungen einerseits und in der Strukturierung der Entscheidungsprozesse derart, dass sich zeitliche Abfolgen begrenzter Entscheidungsvorgänge herauskristallisieren, die jeweils in Form von Entscheidungstabellen beschrieben werden können, andererseits. Hier ist nur schwer erkennbar, wie sich diese Vorgänge auf eine automatisierbare syntaktische Dimension reduzieren lassen.

Weniger problematisch dürfte die Realisierung der Rechnerunterstützung bei der Erstellung jeder Einzeltabelle sein. Regelentwicklung, Verifikation und Minimierung erfolgen nach ausgereiften Methoden, die zu einem hohen Prozentsatz automatisierbar sein dürften, da hier vorwiegend syntaktische Operationen angewendet werden können.

4.3.2 Anwendung graphentheoretischer Verfahren zur rechnergestützten Entwicklung von Entscheidungstabellen

Die Graphentheorie als Zweig der Topologie hat bereits zahlreiche Annwendungsgebiete bei der Untersuchung von Problemen gefunden, die sich in Form von Strukturen abbilden lassen. Zur Analyse und Transformation von als Graph abgebildeten Strukturen steht eine Fülle auch automatisierbarer Verfahren zur Verfügung, von denen sich einige auf Probleme der Entscheidungstabellentechnik anwenden lassen. Als Voraussetzung ist allerdings der Nachweis einer strukturellen Isomorphie zwischen Entscheidungstabelle und Graph erforderlich. Wie zu zeigen sein wird, gelingt dieser Nachweis für das Entscheidungsmatrixformat der Mehrtreffer-Entscheidungstabelle. Es zeichnet sich dann eine Lösungsmöglichkeit für das Zergliederungsverfahren ab.

Die Nutzung graphentheoretischer Methoden in der Entscheidungstabellentechnik steht noch in ihren ersten Anfängen. Da sich hier jedoch eine Möglichkeit bietet, eine ausgereifte Theorie und eine Fülle vorhandener Analyse- und Transformationsalgorithmen zu übernehmen, sollte insbesondere im Rahmen einer Erforschung von Verfahren zur rechnergestützten Entwicklung von Entscheidungstabellen eine intensive Untersuchung der Brauchbarkeit des graphentheoretischen Instrumentariums erfolgen. Die folgenden Ausführungen sollen eine Basis für weiterführende Untersuchungen schaffen.

4.3.2.1 Einige grundlegende Definitionen der Graphentheorie

Bei den im folgenden wiedergegebenen grundlegenden Definitionen der Graphentheorie handelt es sich nur um eine Auswahl von Definitionen, die zum Verständnis der später erläuterten Anwendung des graphentheoretischen Instrumentariums auf Probleme der Erstellung von Entscheidungstabellen erforderlich sind.

Def.: Ein *Graph* G= (V,E) besteht aus einer Menge $V = \{v_1, v_2 \cdots, v_1\}$ Knoten und einer Menge $E = \{e_1, e_2 \cdots, e_m\}$ Kanten, wobei jede Kante durch ein ungeordnetes Paar von Knoten, die Elemente der Menge V sind, beschrieben werden kann.

Für $e_i \in E$ und $v_{j,} v_k \in V$ sowie $e_i = (v_j v_k)$ gilt die Kante e_i als inzident mit den Knoten v_j und v_k; mit allen anderen Knoten aus *V* gilt e_1 als nicht inzident.[1)]

---

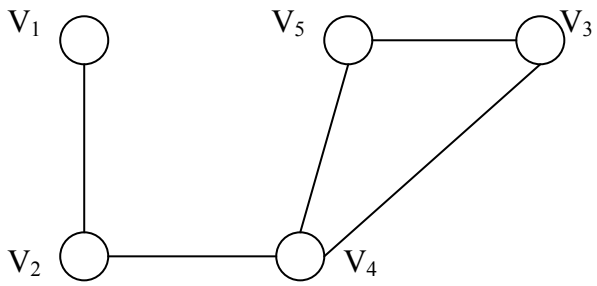
1) Vgl. Harary (Graphentheorie), S. 158

Zur Veranschaulichung eines Graphen wird üblicherweise die Diagramm –Darstellung gewählt. Ein Graph $G = (V, E)$ mit

$$V = \{v_1, v_2, v_3, v_4, v_5\} \text{ und}$$
$$E = \{(v_1, v_2), (v_2, v_4), (v_3, v_4), (v_3, v_5), (v_4, v_5)\}$$

kann wie folgt veranschaulicht werden:

G;



Analysen und Transformationen eines Graphen Lassen sich vielen Fällen einfacher durchführen, wenn die Matrixdarstellung gewählt wird.

Def.: Die *Nachbarmatrix* $N = \lfloor n_{ij} \rfloor$ eines durchnummerierten Graphen mit 1 Knoten ist die (l,l)-Matrix, in welcher $n_{ij} = 1$ gilt, wenn v_i und v_j durch eine Kante verbunden sind und sonst $n_{ij} = 0$ ist.[1)]

Die Nachbarmatrix N (G) ergibt sich dann als (5 x 5)-Matrix:

N (G):

	v_1	v_2	v_3	v_4	v_5
v_1	0	1	0	0	0
v_2	1	0	0	1	0
v_3	0	0	0	1	1
v_4	0	1	1	0	1
v_5	0	0	1	1	0

Wenn Strukturen durch einen Graphen beschrieben werden sollen, bei denen gerichtete Beziehungen zwischen Elementen darstellbar sein müssen, dann werden Digraphen angewendet.

Def.: Ein *gerichteter* Graph oder *Digraph* D = (V,E) ist ein Graph, dessen Kanten aus geordneten Paaren von Knoten bestehen.

Eine Kante $e_i =< v_j, v_k >$ eines Digraphen ist von v_j nach v_k gerichtet.[1)]

Diagramme des Digraphen unterscheiden sich von Diagrammen des Graphen dadurch, dass die Kanten mit Pfeilspitzen versehen sind, die die Verknüpfungsrichtung symbolisieren.

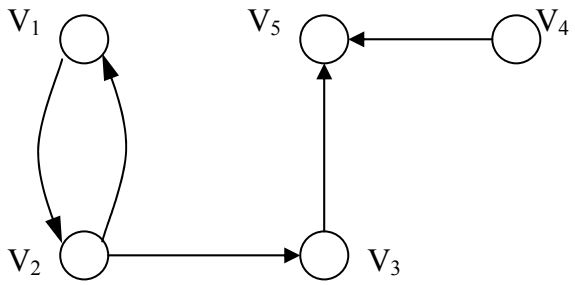
Ein Digraph $D = (V,E)$ mit

$V = \{v_1, v_2, v_3, v_4, v_5\}$ und

$E = \{< v_1, v_2 >, < v_2, v_1 >, < v_2, v_3 >, < v_3, v_5 >, < v_4, v_5 >\}$

Kann durch folgendes Diagramm veranschaulicht werden:

D:



Die Nachbarmatrix des Digraphen wird zutreffender als "Präzedenzmatrix" bezeichnet.[2)] Damit wird ausgedrückt, dass aus gerichteten Beziehungen Präzedenzen der Knoten resultieren.

Def.: Die *Präzedenzmatrix* $P(D) = \lfloor P_{ij} \rfloor$ des Digraphen D mit 1 Knoten ist die (l,l)-Matrix mit $P_{ij} = 1$, wenn <v_i, v_j> eine Kante von D ist und sonst $P_{ij} = 0$.[3)]

Zum verwendeten Beispiel lässt sich folgende Präzedenzmatrix erstellen:

P (D):

	v_1	v_2	v_3	v_4	v_5
v_1	0	1	0	0	0
v_2	1	0	1	0	0
v_3	0	0	0	0	1
v_4	0	0	0	0	1
v_5	0	0	0	0	0

1) Zur Definition des Digraphen vgl. die ausführlichen Darstellungen bei Harary; Norman; Cartwright (Structural models), S. 9 f.
2) Vgl. so z.B. bei Langefors (Information systems), S. 98
3) Vgl. Harary (Graphentheorie), S. 159

Die Zeilen- und Spaltensummen der Präzedenzmatrix zeigen zu jedem Knoten die Anzahl der nach außen und nach innen gerichteten Kanten an.

Def.: Als *Außengrad* eines Knotens v gilt die Anzahl der von diesem Knoten ausgehenden Kanten.
Als *Innengrad* eines Knotens v gilt die Anzahl der in diesen Knoten eingehenden Kanten.[1)]

Wenn Digraphen mit gleicher Knotenzahl miteinander verglichen werden, kann trotz unterschiedlicher Diagramme oder Präzedenzmatrizen Isomorphie bestehen.

Satz: Zwei Digraphen D_1 und D_2 sind dann und nur dann isomorph, wenn für jede Anordnung der Knoten von D_1 eine Anordnung der Knoten von D_2 so gefunden werden kann, dass die Präzedenzmatrizen P_1 und P_2 identisch sind.[2)]

Neben der Präzedenzmatrix wird für viele graphentheoretische Transformationen die Erreichbarkeitsmatrix benötigt, die aus der Präzedenzmatrix abgeleitet werden kann.[3)]

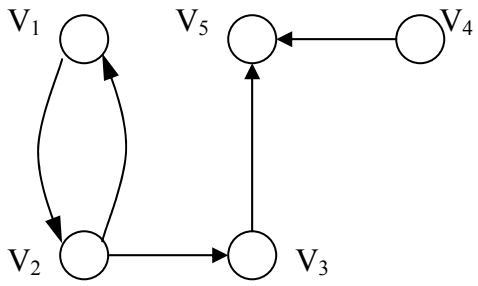
Def.: Die *Erreichbarkeitsmatrix* $R\lfloor r_{ij} \rfloor$ eines durchnummerierten Digraphen mit 1 Knoten ist die (l,l)-Matrix, in welcher $r_{ij} = 1$ gilt, wenn v_j von v_i aus erreichbar ist und $r_{ij} = 0$ in allen anderen Fällen. Ein Knoten V_j ist von einem Knoten V_i genau dann erreichbar, wenn zwischen beiden ein Weg aus einer nach V_j . gerichteten Kantenfolge nachgewiesen werden kann.[4)] Jeder Knoten gilt als von sich selbst aus erreichbar.[5)]

---

1) Vgl. Harary; Norman; Cartwright (Structural models), S. 16
2) Zum Beweis dieses Satzes vgl. Harary; Norman; Cartwright (Structural models), S. 20 f.
3) Vgl. Harary (Graphentheorie), S. 213
4) Zur ausführlichen Erläuterung der Begriffe "Weg" und "Kantenfolge" vgl. Harary (Graphentheorie), S. 208
5) Vgl. Harary; Norman; Cartwright (Structural models), S. 117

Der als Beispiel verwendete Digraph hat folgende Erreichbarkeitsmatrix:

D:



R (D):

	v_1	v_2	v_3	v_4	v_5
v_1	0	1	0	0	0
v_2	1	0	0	1	0
v_3	0	0	0	1	1
v_4	0	1	1	0	1
v_5	0	0	1	1	0

Digraphen können nach der Art der Verknüpfung ihrer Knoten klassifiziert werden.

Def.: Ein Digraph ist *stark zusammenhängend*, wenn beliebige zwei seiner Knoten gegenseitig erreichbar sind. Auch ein Digraph, der nur einen Knoten enthält, gilt als stark zusammenhängend.

Ein Digraph ist *unilateral zusammenhängend*, wenn von jeweils zwei seiner Knoten mindestens einer vom anderen aus erreichbar ist.

Ein Digraph ist *schwach zusammenhängend*, wenn jeweils zwei seiner Knoten durch Kanten verbunden sind.

Ein Digraph ist *unzusammenhängend*, wenn er noch nicht einmal schwach zusammenhängend ist. [1)]

Die definierten Eigenschaften lassen sich auch Subgraphen zuordnen. Jeder Subgraph S eines Digraphen *D* besteht aus einer Teilmenge der Knoten von *D* und den diese verbindenden Kanten.[2)]

Def.: Ein stark zusammenhängender Subgraph eines Digraphen ist eine *starke Komponente* dieses Digraphen. Ein unilateral zusammenhängender Subgraph ist eine *unilaterale Komponente*. Ein schwach zusammenhängender Subgraph ist eine *schwache Komponente*.[3)]

Auf der Basis seiner starken Komponenten lässt sich ein Digraph strukturell verdichten.

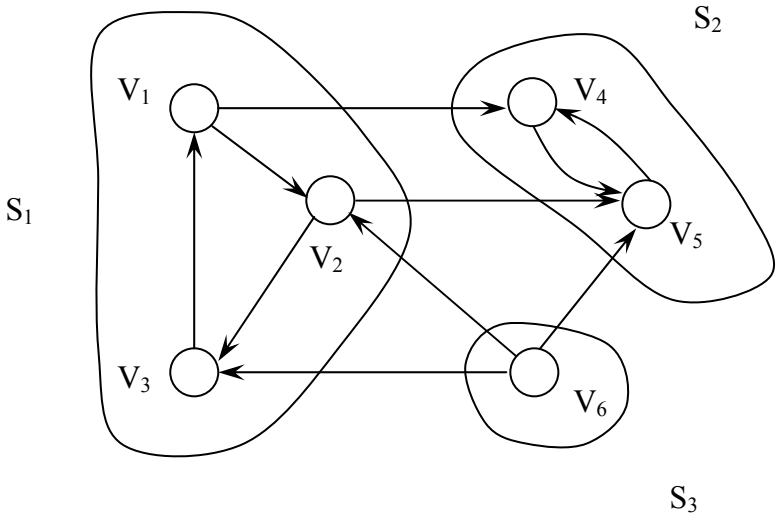
---

1) Vgl. Harary Norman; Cartwright (Structural models), S. 51
2) Vgl. Harary Norman; Cartwright (Structural models), S. 53 f.
3) Vgl. Harary Norman; Cartwright (Structural models), S. 54

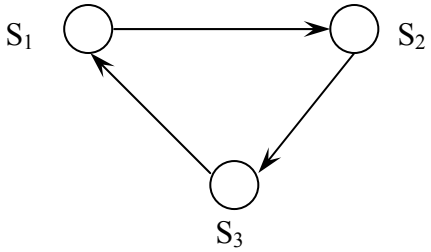
Def.: Als *Kondensationsgraph* bezeichnet man einen Digraph, dessen starke Komponenten S jeweils durch einen Knoten ersetzt wurden. [1)]

Beispiel: [2)]

Diagraph:



Kondensationsgraph:



Die Knoten des Kondensationsgraphen werden dann durch Kanten verbunden, wenn vorher mindestens jeweils einer der in einer starken Komponente enthaltenen Knoten mit einem in der anderen starken Komponente enthaltenen Knoten verbunden war.

Auf der Basis seiner Wege lässt sich ein Digraph als zyklisch oder azyklisch klassifizieren.

Def.: Enthält ein Digraph mindestens eine Kantenfolge, die einen Zyklus beschreibt, dann gilt der ganze Digraph als *zyklisch*.

In einem Zyklus ist jeder enthaltene Knoten von jedem anderen Knoten des Zyklus erreichbar.[3)]

---

1) Vgl. Harary Norman Cartwright (Structural models), S. 57 ff.
2) Vgl. Harary Norman Cartwright (Structural models), S. 60
3) Vgl. Harary Norman Cartwright (Structural models), S. 40 ff.

Def.: Ein Digraph, der keinen Zyklus enthält, gilt als *azyklischer* Digraph.[1)]

Ein zyklischer Digraph lässt sich demnach durch Entfernung seiner Zyklen in einen azyklischen Digraphen verwandeln. Jeder Kondensationsgraph eines Digraphen ist somit azyklisch. Jeder Kondensationsgraph eines azyklischen Digraphen ist zum ursprünglichen Digraphen isomorph.[2)]

4.3.2.2 Entscheidungstabelle und Entscheidungstabellenverbund als Graph

Graphen eignen sich zur Abbildung von Systemen, die aus Elementen sowie aus Beziehungen zwischen diesen Elementen bestehen. In stark vereinfachter Betrachtungsweise lässt sich auch der durch eine Entscheidungstabelle oder einen Entscheidungstabellenverbund abgebildete Sachverhalt formal als ein solches System von Elementen und Beziehungen zwischen den Elementen auffassen. Die zu prüfenden Bedingungen sowie die auszuführenden Aktionen werden dabei als Elemente, die logischen Beziehungen zwischen Bedingungen und Aktionen als Beziehungen zwischen den Elementen angesehen.

Es wäre jedoch schwierig, aus einer Eintreffer-Entscheidungstabelle die Beziehungen zwischen Bedingungen und Aktionen abzulesen. Da innerhalb jeder Entscheidungsregel *Bedingungskonstellationen* und *Aktionsfolgen* abgebildet sind, bleiben die Beziehungen jeder enthaltenen Bedingung zu jeder enthaltenen Aktion und vice versa verborgen. Erst nach Umwandlung der Eintreffer-Entscheidungstabelle in eine Mehrtreffer-Entscheidungstabelle im Entscheidungsmatrixformat[3)] werden die logischen Beziehungen zwischen Bedingungen und Aktionen transparent.

Für bestimmte syntaktische Transformationen der Entscheidungstabelle, beispielsweise für deren Zergliederung, ist es ausreichend zu wissen, dass eine Bedingung eine Aktion beeinflusst. Nicht erforderlich ist die Kenntnis des Bedingungs*zustande*s, der als Voraussetzung für die Ausführung der Aktion gilt.[4)] Dann ist aber eine Darstellung einer Entscheidungstabelle in Form eines Digraphen möglich, wie folgendes Bei-

---

1) Vgl. Harary; Norman; Cartwright (Structural models), S. 63
2) Vgl. Harary; Norman; Cartwright (Structural models), S. 63
3) Die dazu erforderlichen Transformationen sind bereits in den Abschnitten 2.1.2.2 und 2.1.2.3 ausführlich diskutiert worden.
4) Vgl. Strunz (Parsing), S. 367

spiel illustriert: [1)]

Eintreffer-Entscheidungstabellen:

		1	2	3	4
B1	Bedingung a	J	J	J	J
B2	Bedingung b	J	J	N	N
B3	Bedingung c	J	N	-	-
B4	Bedingung d	-	-	J	-
B5	Bedingung e	-	-	J	N
A1	Aktion α	X	X		
A2	Aktion β	X		X	
A3	Aktion γ				X
A4	Aktion δ	X	X		X

Entscheidungsmatrix:

	A1		A2		A3	A4		
B1	J	J	J	J	J	J	J	J
B2	J	J	J	N	N	J	J	N
B3	J	N	J	-	-	J	N	-
B4	-	-	-	J	-	-	-	-
B5	-	-	-	J	N	-	-	N

Minimierte Entscheidungsmatrix:

	A1	A2		A3	A4	
B1	J	J	J	J	J	J
B2	J	J	N	N	J	-
B3	-	J	-	-	-	-
B4	-	-	J	-	-	-
B5	-	-	J	N	-	N

Reduzierte Entscheidungsmatrix: Es wird nur noch durch "X" angezeigt, dass die in der Zeile aufgeführte Bedingung die in der Spalte bezeichnete Aktion beeinflusst:

---

1) Vgl. Strunz ; Jorgensen (Anwendung graphentheoretischer Verfahren), S. 65 f.

	A1	A2	A3	A4
B1	X	X	X	X
B2	X	X	X	X
B3		X		
B4		X		
B5		X	X	X

Bei Betrachtung der Bedingungen und Aktionen als Knoten und der logischen Beziehungen als Kanten lässt sich die reduzierte Entscheidungsmatrix unmittelbar als Digraph interpretieren, dessen Präzedenzmatrix folgende Form hat:

	A1	A2	A3	A4
B1	1	1	1	1
B2	1	1	1	1
B3	0	1	0	0
B4	0	1	0	0
B5	0	1	1	1

Logisch gleichwertig wäre die quadratische Darstellung mit 9 x 9 Matrixelementen:

	B1	B2	B3	B4	B5	A1	A2	A3	A4
B1	0	0	0	0	0	1	1	1	1
B2	0	0	0	0	0	1	1	1	1
B3	0	0	0	0	0	0	1	0	0
B4	0	0	0	0	0	0	1	0	0
B5	0	0	0	0	0	0	1	1	1
A1	1	1	0	0	0	0	0	0	0
A2	1	1	1	1	1	0	0	0	0
A3	1	1	0	0	1	0	0	0	0
A4	1	1	0	0	1	0	0	0	0

In ihrer quadratischen Darstellung eröffnet die Präzedenzmatrix die Möglichkeit, zusätzliche semantische Charakteristika der Entscheidungstabelle darzustellen. Da sich auch "1"-Elemente für Beziehungen der Bedingungen untereinander und der Aktionen miteinander eintragen lassen, können semantisch bedeutungsvolle Reihenfolge-

forderungen für die Anordnung von Bedingungen und Aktionen spezifiziert werden. Angenommen, aufgrund logischer Abhängigkeiten der Bedingungen darf die Bedingung 5 keinesfalls vor den Bedingungen 1 - 4 geprüft werden. Außerdem soll sichergestellt werden, dass die Aktionen in der Reihenfolge A1-A2-A3-A4 ausgeführt werden. Die entsprechende Präzedenzmatrix ergibt sich dann wie folgt:[1)]

	B1	B2	B3	B4	B5	A1	A2	A3	A4
B1	0	0	0	0	(1)	1	1	1	1
B2	0	0	0	0	(1)	1	1	1	1
B3	0	0	0	0	(1)	0	1	0	0
B4	0	0	0	0	(1)	0	1	0	0
B5	0	0	0	0	0	0	1	1	1
A1	1	1	0	0	0	0	(1)	0	0
A2	1	1	1	1	1	0	0	(1)	0
A3	1	1	0	0	1	0	0	0	(1)
A4	1	1	0	0	1	0	0	0	0

Logische Beziehungen zwischen Bedingungen und Aktionen, die sich nicht aus den Entscheidungsregeln ableiten lassen, bleiben bei der bisher gewählten Darstellung unberücksichtigt. Beispielsweise könnte eine Aktion eine Variable verändern, die bei nochmaliger Ausführung der Entscheidungstabelle als Bedingung die Regelauswahl beeinflusst. Wenn die Änderungsaktion selbst jedoch unabhängig vom Zustand der Variable ist, würde in der Entscheidungsmatrix keine logische Beziehung zwischen Bedingung und Aktion ausgewiesen.

Durch die Aufnahme von *Datenelementen* D_i in die Präzedenzmatrix und Darstellung ihrer Beziehungen zu Bedingungen und Aktionen lässt sich dieses Problem lösen. Durch "1" als Matrixelement wird angezeigt, ob ein Datenelement Bestandteil einer Bedingungsprüfung ist oder durch eine Aktionsausführung beeinflusst wird. Werden zu den vorhandenen 5 Bedingungen und 4 Aktionen noch 3 Datenelemente ergänzt, so ergibt sich eine (5+4+3) x (5+4+3)-Matrix, deren Elemente die Darstellung folgender Beziehungen erlauben:2)

---

1) Vgl. Strunz; Jorgensen (Anwendung graphentheoretischer Verfahren), S. 66
2) Vgl. Strunz; Jorgensen (Anwendung graphentheoretischer Verfahren), S. 66

	B1 B2 B3 B4 B5	A1 A2 A3 A4	D1 D2 D3
B1 B2 B3 B4 B5	I Beziehungen der Bedingungen untereinander	II Beziehungen der Bedingung zu den Aktionen	III Beziehungen der Bedingungen zu den Daten
A1 A2 A3 A4	Transponierte Matrix zu II	IV Beziehungen der Aktionen untereinander	V Beziehungen der Aktionen zu den Daten
D1 D2 D3	Transponierte Matrix zu III	Transponierte Matrix zu V	VI Beziehungen der Daten untereinander

Die in dieser Präzedenzmatrix ebenfalls eröffnete syntaktische Möglichkeit, Beziehungen der Datenelemente untereinander darzustellen, kann durchaus auch semantisch sinnvoll genutzt werden. Beispielsweise könnte durch eine gerichtete Beziehung zwischen D1 und D2 angezeigt werden, dass zur Erarbeitung von D2 D1 als Input benötigt wird.[1)] .

Zur Darstellung eines *Entscheidungstabellenverbundes* als Digraph werden die Einzeltabellen des Verbunds als Knoten des Digraphen definiert. Diese Einzeltabellen können untereinander prozedural oder dateninduziert verknüpft sein.[2)]

Drei elementare *prozedurale* Verknüpfungen lassen sich unterscheiden: Die Sequenz, die Iteration und die Verschachtelung. Eine gerichtete Kante zwischen zwei Knoten kann als sequentielle Verknüpfung der repräsentierten Tabellen interpretiert werden. Eine zyklische Kantenfolge zwischen mehreren Knoten kennzeichnet die Iteration und die direkte zweiseitige Verbindung von zwei Knoten die Verschachtelung. Als Beispiel soll der im Folgenden mit Hilfe eines Diagramms veranschaulichte Entscheidungstabellenverbund als Digraph definiert wer-

1) Vgl. dazu auch die Darstellung von Präzedenzbeziehungen zwischen Informationen mit Hilfe der Präzedenzmatrix bei Langefors (Information systems), S. 283

2) Vgl. Strunz (Decision table networks), S. 1 ff.

den, wobei die Matrixdarstellung benutzt werden soll.[1)]

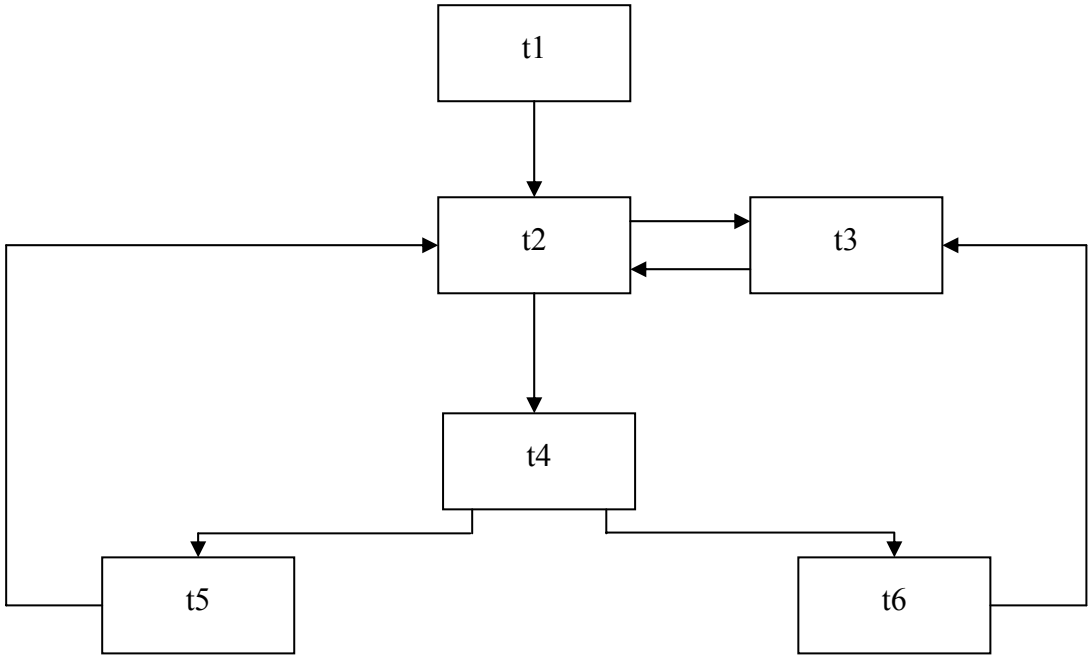


Abb. 6: Diagrammdarstellung eines Entscheidungstabellenverbundes

Digraph der Tabellen und ihrer prozeduralen Beziehungen in der Darstellung seiner Präzedenzmatrix:

	t1	t2	t3	t4	t5	t6
t1	0	1	0	0	0	0
t2	0	0	1	1	0	0
t3	0	1	0	0	0	0
t4	0	0	0	0	1	1
t5	0	1	0	0	0	0
t6	1	0	0	0	0	0

Elementare *dateninduzierte* Verknüpfungen von Entscheidungstabellen eines Verbunds sind die Verknüpfung durch Übergabe lokaler Variablen und die Verknüpfung durch gemeinsame Benutzung globaler Variablen. Seien *A, B*, ... , *K* Variablen, die innerhalb des als Beispiel verwendeten Entscheidungstabellenverbunds benutzt werden. Es soll nun angenommen werden, dass in den Einzeltabellen die folgenden Variablen benutzt werden:

---

1) Vgl. Strunz; Jorgensen (Anwendung graphentheoretischer Verfahren), S. 67

t1	:	A,	B,	H
t2	:	C,	D,	E
t3	:	D,	K	
t4	:	A,	I	
t5	:	D,	F,	J
t6	:	E,	G	

C sei eine lokale Variable, die durch Übergabe der Variablen *B* von t_1 an t_2 entstanden ist. *F* sei eine lokale Variable, die durch Übergabe der Variablen *I* von t_4 nach t_5 entstanden ist. *G* sei eine lokale Variable, die durch Übergabe der Variablen *I* von t_4 nach t_6 entstanden ist. Wenn die Übergabe einer lokalen Variablen durch eine einseitig gerichtete Kante und die Benutzung einer globalen Variable durch eine zweiseitig gerichtete Kante dargestellt wird, dann ergibt sich der Graph der Tabellen mit ihren dateninduzierten Verknüpfungen wie folgt:[1)]

	t1	t2	t3	t4	t5	t6
t1	0	1	0	1	0	0
t2	0	0	1	0	1	1
t3	0	1	0	0	0	0
t4	1	0	0	0	1	1
t5	0	1	0	0	0	0
t6	0	1	0	0	0	0

4.3.2.3 Graphentheoretische Lösung des Zergliederungs- und Modularisierungsproblems

Mit zunehmender Anzahl der Bedingungen steigt die Anzahl formal möglicher Bedingungskonstellationen rasch an. Da aus pragmatischen Überlegungen heraus die Anzahl der Entscheidungsregeln einer Entscheidungsstabelle eine bestimmte Grenze, die oft mit ca. 30 angegeben wird, nicht übersteigen sollte, entsteht häufig das Problem, umfangreiche Mengen von Bedingungen und Aktionen so zu zergliedern, dass kleinere Bedingungs-/ Aktionsgruppen entstehen. Da jede dieser Bedingungs-/Aktionsgruppen in eine Entscheidungstabelle überführbar sein soll, lässt sich die Zielsetzung für diese Zergliederung wie folgt definieren:

---

1) Vgl. Strunz; Jorgensen (Anwendung graphentheoretischer Verfahren), S. 67

(1) Die Anzahl der Bedingungen jeder Gruppe muss auf eine Annzahl limitiert werden, die sicherstellt, dass die zu erwartende Regelanzahl ein akzeptiertes Maß nicht übersteigt.

(2) Die verschiedenen Bedingungs-/Aktionsgruppen sollen so definiert werden, dass die logischen Beziehungen der Gruppen zueinander *minimiert* werden. Zu identischen Ergebnissen führt die Forderung, die Gruppen so zu definieren, dass die logischen Beziehungen der Bedingungen und Aktionen innerhalb jeder Gruppe *maximiert* werden.

Ein formal identisches Zergliederungsproblem stellt sich bei der Modularisierung eines zunächst monolitisch aufgebauten Entscheidungsstabellenverbunds. Auch dabei muss die Zielsetzung verfolgt werden, die Einzeltabellen des Verbunds so auf verschiedene Moduln aufzuteilen, dass die Moduln eine bestimmte akzeptierte Größe nicht übersteigen und die logischen Beziehungen der Moduln zueinander minimiert bzw. die logischen Beziehungen der Einzeltabellen innerhalb jedes Moduls maximiert werden.

Ein Lösungsansatz für das Zergliederungsproblem[1)], der auf der Basis der logischen Beziehungen zwischen Bedingungen und Aktionen eine manuelle Zergliederung ermöglicht, wurde bereits diskutiert.[2)]

Graphentheoretische Verfahren ermöglichen eine automatische Zergliederung. Wie bereits gezeigt wurde, lassen sich eine Entscheidungsstabelle oder ein Entscheidungstabellenverbund als Digraph formulieren. Das Zergliederungsproblem kann dann durch Bestimmung der starken Komponenten des Entscheidungstabellen -Digraphen oder des Entscheidungstabellenverbund -Digraphen gelöst werden. Die Bestimmung starker Komponenten erfolgt mit Hilfe der Erreichbarkeitsmatrix.

Sei *P* die Präzedenzmatrix eines Digraphen, *E* die Einheitsmatrix und k die Anzahl der Knoten des Digraphen, so ergibt sich die Erreichbarkeitsmatrix *R* wie folgt:[3)]

(1) $R=E+P+P^2 + ... +P^{k-1}$;
(2) alle positiven Matrixelemente werden auf "1" gesetzt.

---

1) Vgl. Strunz (Zergliederung), S. 1 ff., und Strunz (Parsing), S. 367 ff.
2) Vgl. dazu Abschnitt 1.1.2.2.1
3) Vgl. dazu Harary; Norman; Cartwright (Structural models), S. 121 f.
Harary, Norman und Cartwright schlagen vor, zur Addition der Matrixelemente mit dem logischen UND zu arbeiten. Dann entfällt Schritt (2).

Die angegebene maximale Anzahl der Summanden braucht nicht immer berechnet zu werden. Wenn ein p^i nur noch Nullelemente aufweist, kann auf die Bildung von p^{i+l} bis p^{k-l} verzichtet werden, da diese ebenfalls nur noch Nullelemente enthalten würden.

Zur Ermittlung der starken Komponenten aus der Erreichbarkeitsmatrix ist dann nur noch die Multiplikation von R mit ihrer transponierten Form R' erforderlich.[1)]

Die automatische Zergliederung unter Verwendung des erläuterten Verfahrens soll an einem Beispiel verdeutlicht werden, das auch bereits zur Veranschaulichung des manuellen Zergliederungsverfahrens gewählt wurde.[2)]

Zu einem beliebigen Problem seien 8 Bedingungen und 11 Aktionen ermittelt worden, deren logische Beziehungen in Form der folgenden reduzierten Entscheidungsmatrix beschrieben wurden:

	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11
B1			X	X	X			X	X		
B2	X	X									
B3			X		X			X	X		
B4						X	X				
B5										X	X
B6			X	X	X			X	X		
B7										X	
B8		X									

Aus dieser Vorgabe kann leicht die (8+11) x (8+11)-Präzedenzmatrix gebildet werden.

---

1) Vgl. dazu Harary; Norman; Cartwright (Structural models), S. 123 ff.
2) Vgl. dazu Strunz (Parsing), S. 368

	B1	B2	B3	B4	B5	B6	B7	B8	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11
B1	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	1	1	0	0
B2	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0
B3	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	1	1	0	0
B4	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0
B5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
B6	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	1	1	0	0
B7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
B8	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
A1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
A2	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
A3	1	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
A4	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
A5	1	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
A6	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
A7	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
A8	1	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
A9	1	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
A10	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0
A11	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Die automatische Ermittlung der starken Komponenten führt zu folgendem Ergebnis:[1)]

1) Vgl. Strunz; Jorgensen (Anwendung graphentheoretischer Verfahren), S. 68
Es wurde ein Softwarepaket zu graphentheoretischen Verfahren der GTE Automatic Electric Laboratories, Chicago, benutzt.

```
V ( 1)     1  1  1  1  1  1  1  1  0  0  0  0  0  0  0  0  0  0  0  0
V ( 3)     1  1  1  1  1  1  1  1  0  0  0  0  0  0  0  0  0  0  0  0
V ( 6)     1  1  1  1  1  1  1  1  0  0  0  0  0  0  0  0  0  0  0  0
V (11)     1  1  1  1  1  1  1  1  0  0  0  0  0  0  0  0  0  0  0  0
V (12)     1  1  1  1  1  1  1  1  0  0  0  0  0  0  0  0  0  0  0  0
V (13)     1  1  1  1  1  1  1  1  0  0  0  0  0  0  0  0  0  0  0  0
V (16)     1  1  1  1  1  1  1  1  0  0  0  0  0  0  0  0  0  0  0  0
V (17)     1  1  1  1  1  1  1  1  0  0  0  0  0  0  0  0  0  0  0  0
V ( 9)     0  0  0  0  0  0  0  0  1  1  1  1  0  0  0  0  0  0  0  0
V (10)     0  0  0  0  0  0  0  0  1  1  1  1  0  0  0  0  0  0  0  0
V ( 2)     0  0  0  0  0  0  0  0  1  1  1  1  0  0  0  0  0  0  0  0
V ( 8)     0  0  0  0  0  0  0  0  1  1  1  1  0  0  0  0  0  0  0  0
V ( 4)     0  0  0  0  0  0  0  0  0  0  0  0  1  1  1  0  0  0  0
V (14)     0  0  0  0  0  0  0  0  0  0  0  0  1  1  1  0  0  0  0
V (15)     0  0  0  0  0  0  0  0  0  0  0  0  1  1  1  0  0  0  0
V ( 7)     0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  1  1  1
V ( 5)     0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  1  1  1
V (18)     0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  1  1  1
V (19)     0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  1  1  1

                    COMPONENT NUMBER   1
                                       C ( 1)
                                       C ( 3)
                                       C ( 6)
                                       A ( 3)
                                       A ( 4)
                                       A ( 5)
                                       A ( 8)
                                       A ( 9)
                    COMPONENT NUMBER   2
                                       A ( 1)
                                       A ( 2)
                                       C ( 2)
                                       C ( 8)
                    COMPONENT NUMBER   3
                                       C ( 4)
                                       C ( 6)
                                       C ( 7)
                    COMPONENT NUMBER   4
                                       C ( 7)
                                       C ( 5)
                                       A(10)
                                       A(11)
                         STOP
                         PROCESSING    62 UNITS
```

Abb. 7: Ergebnis der Ermittlung starker Komponenten

Unter der Voraussetzung, dass die Semantik des Problems es erfordert, die Aktionen in der innerhalb der Entscheidungsmatrix angegebenen Reihenfolge auszuführen, lässt sich aus den ermittelten starken Komponenten folgender Entscheidungstabellenverbund erstellen:

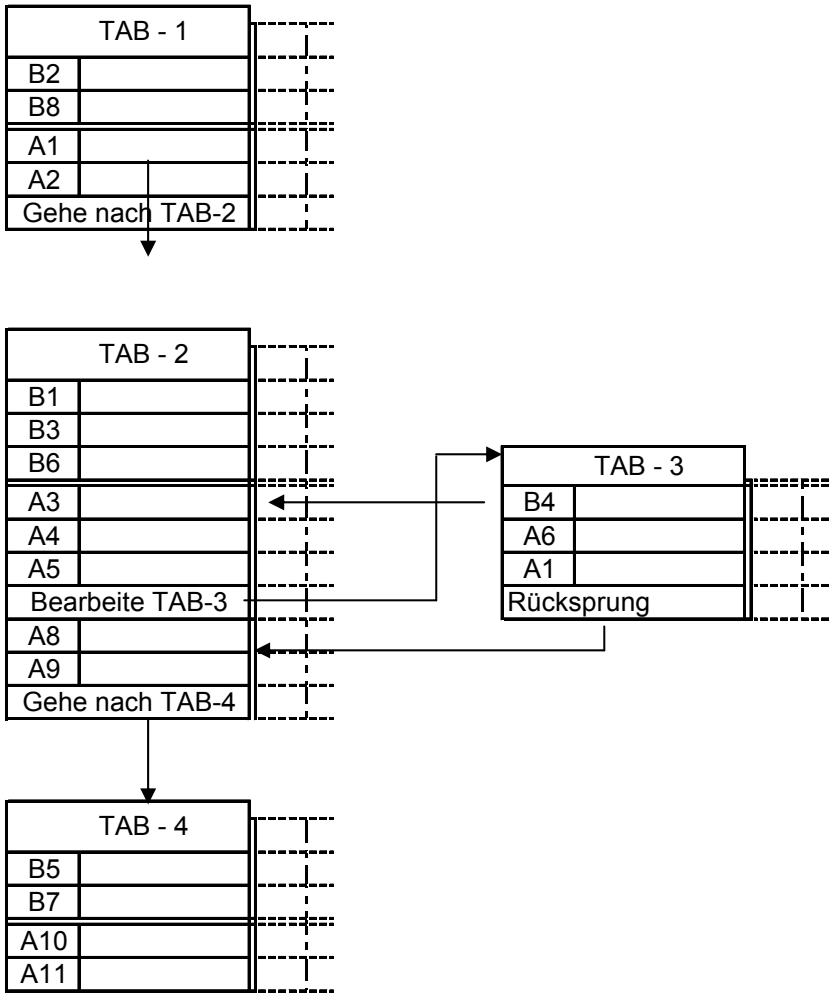


Abb. 8: Entscheidungstabellenverbund nach Zergliederung

Das entspricht dem Ergebnis der Anwendung des manuellen Zergliederungssverfahrens.[1)]

Bei intensiveren logischen Beziehungen zwischen Bedingungen und Aktionen kann der Fall auftreten, dass keine starken Komponenten ermittelt werden können oder einige der entstehenden starken Komponenten noch zu umfassend, d.h. die dadurch abgegrenzten Bedingungs-/Aktionsgruppen zu umfangreich sind zur unmittelbaren Übertragung in eine Entscheidungsstabelle. Auch in diesem Fall bietet jedoch die Graphentheorie einen Lösungsansatz. Jeder Digraph kann einer Analyse unterzogen werden, die die minimale Schnittmenge der Knoten ermittelt.[2)] Wenn beispielsweise die minimale Schnittmenge der Knoten durch eine einzige Bedingung gebildet wird, dann führt die Entfernung dieser Bedingung zu einer Aufgliederung des Digraphen in mindestens zwei Komponenten. Bei der Gestaltung der aus den Komponenten entstehenden Entscheidungstabellen wird die entfernte Bedingung wieder ergänzt. Sie wird Bestandteil *jeder* der aus der ursprünglichen umfassenden starken Komponente resultierenden Teiltabelle. Auch wenn die Schnittmenge aus zwei oder drei Bedingungen besteht, kann eine Aufgliederung und eine damit unvermeidliche mehrfache Platzierung von Bedingungen sinnvoll sein. Eine starke Komponente, die beispielsweise 10 Bedingungen mit jeweils zwei Zuständen enthält, eröffnet die Möglichkeit zur Beschreibung einer Entscheidungslogik für 2^{10} = 1024 Bedingungskonstellationen, was in aller Regel die Möglichkeiten einer einzelnen Entscheidungstabelle übersteigt. Wenn die minimale Schnittmenge der Knoten aus 3 Bedingungen gebildet wird und nach ihrer Entfernung zwei Komponenten mit 3 und 4 Bedingungen entstehen, dann lassen sich zwei Entscheidungstabellen erstellen, von denen die eine 6 Bedingungen und die andere 7 Bedingungen enthält.

Formal bieten diese beiden Entscheidungstabellen die Möglichkeit zur Beschreibung von $2^6 + 2^7 = 192$ Bedingungskonstellationen. Die Wahrscheinlichkeit, dass diese Anzahl Bedingungskonstellationen mit den Regeln zweier Tabellen abgedeckt werden kann, ist hoch.

Die automatische Ableitung starker Komponenten aus einer Präzedenzmatrix, die nur die logischen Beziehungen zwischen Bedingungen und Aktionen berücksichtigt, lässt sich noch effizienter durchführen, wenn die Ableitung

---

1) Vgl. Strunz (Parsing), S. 368

2) Zur minimalen Schnittmenge der Knoten vgl. Harary; Norman; Cartwright (Structural models), S. 229 ff. und S. 237 ff.

nicht auf der Basis der vollständigen Präzedenzmatrix, sondern einer Matrix erfolgt, deren Zeilenelemente nur aus den Bedingungen und deren Spaltenelemente nur aus den Aktionen bestehen.[1)] Die Matrixgröße reduziert sich dann für das erläuterte Zergliederungsbeispiel von (8+ll) x (8+11) = 361 auf 8 x 11 = 88 Elemente.

Die vollständige Präzedenzmatrix erlaubt jedoch die zusätzliche Einbeziehung semantisch relevanter Merkmale des Problems. Wie bereits gezeigt wurde, lassen sich semantische Reihenfolgeforderungen für Bedingungen und Aktionen in der Matrix spezifizieren,[2)] die als Nebenbedingungen für den Zergliederungsprozess gelten können.

Ebenso, wie zu große Entscheidungstabellen aus pragmatischer Sicht ungünstig zu beurteilen sind, können jedoch auch zu kleine Entscheidungstabellen, die aus einer konsequenten Zergliederung resultieren, eine Problembeschreibung zu sehr zerstückeln und einige der Vorteile der Anwendung von Entscheidungstabellen wieder neutralisieren. Eine Lösung dieses Problems liegt möglicherweise in der Aufnahme der von den Bedingungen und Aktionen benutzten Variablen in den Zergliederungsvorgang.[3)] Wenn die Ableitung starker Komponenten unter Einbeziehung gemeinsamer Variablen erfolgt, dann können nicht mehrere Teiltabellen identische Variablen benutzen. Wenn jedoch die Bearbeitungsvorgänge für die unterschiedlichen Variablen in einzelnen Entscheidungstabellen konzentriert sind, wird die Problembeschreibung übersichtlicher.

Der Gefahr der Entstehung zu großer Tabellen kann wieder durch eine Analyse der Knoten-Schnittmengen begegnet werden. Es können so schrittweise einzelne Variablen wieder zur Bearbeitung in mehreren Tabellen freigegeben werden, bis die entstehenden Tabellengrößen der Zielsetzung entsprechen.

Die Ableitung starker Komponenten kann auch zur Modularisierung eines Entscheidungstabellenverbunds dienen.[4)] Soll sich die Zergliederung nur an den prozeduralen Beziehungen der Einzeltabellen des Verbunds orientieren, dann werden die starken Komponenten aus Tabellen gebildet, die untereinander in iterativer oder verschachtelter Beziehung stehen.

---

1) Vgl. Harary; Norman; Cartwright (Structural models), S. 323.
In der angelsächsischen Terminologie zur Graphentheorie wird eine solche Matrix "bipartite" genannt.
2) Vgl. Abschnitt 4.3.2.2
3) Die Darstellung der Variablen innerhalb des Graphen ist im Abschnitt 4.3.2.2 erläutert worden.
4) Vgl. Strunz; Jorgensen (Anwendung graphentheoretischer Verfahren), S. 70

Werden auch die dateninduzierten Beziehungen in den Modularisierungsvorgang einbezogen, dann werden alle Tabellen zu jeweils einem Modul zusammengefasst, die entweder iterativ oder verschachtelt prozedural verknüpft sind und/oder gemeinsame Variablen benutzen. Die logischen Beziehungen unterschiedlicher Moduln beschränken sich dann auf sequentielle prozedurale Beziehungen und die Übergabe lokaler Variablen.

Grundlage der Modularisierung eines Entscheidungstabellenverbunds mit Hilfe graphentheoretischer Methoden sind die in Abschnitt 4.3.2.2 definierten Digraphen, in deren Matrix-Wiedergabe prozedurale oder dateninduzierte Verknüpfungen durch "1" angezeigt sind. Es empfiehlt sich, zunächst die starken Komponenten des Digraphen der prozeduralen Beziehungen zu bilden. Danach werden die starken Komponenten des Digraphen der dateninduzierten Beziehungen gebildet. Die starken Komponenten des Digraphen der prozeduralen Beziehungen werden dann stufenweise so lange kombiniert, bis keine starke Komponente mehr zu den starken Komponenten des Graphen der dateninduzierten Beziehungen in Widerspruch steht. Sollte durch diese Kombination ein Modul die beabsichtigte Größe überschreiten, dann empfiehlt sich die gezielte Umwandlung globaler Variablen in lokale Variablen.

4.3.3 Rechnergestützte Erstellung und Verifikation von Einzeltabellen

Im Gegensatz zum Problem, ein komplexes Entscheidungsfeld so zu strukturieren, dass Bedingungs-/Aktionsgruppen abgegrenzt werden können, die jeweils in eine einzige Entscheidungstabelle überführt werden können, erscheint die rechnergestützte Erstellung von Einzeltabellen als methodisch weniger anspruchsvoll. Während zur Strukturierung von Entscheidungsfeldern mit den erläuterten graphentheoretischen Verfahren nur die ersten Ansätze einer rechnergestützten Vorgehensweise formuliert sind, kann die rechnergestützte Erstellung von einzelnen Entscheidungstabellen auf syntaktischer Grundlage als methodisch und algorithmisch gelöst und die Implementierung entsprechender Verfahren als softwaretechnisch leicht realisierbar gelten. Bei der rechnergestützten Erstellung von Entscheidungstabellen können alle syntaktisch durchführbaren Aufbau- und Transformationsoperationen automatisch erfolgen; semantische Informationen müssen manuell erstellt und in Form vorgefertigter syntaktischer Einheiten in den Rechner eingegeben werden. Bereits das in Abschnitt 1.2.4.2 erläuterte Verfahren zur manuellen Erstellung von Entscheidungstabellen zeigte jedoch die Notwendigkeit des ständigen Wechsels einer Fixierung semantischer Informationen mit der Ausführung syntaktischer Aufbau- und Transformationsoperationen. Eine rechnergestützte Erstellung von Entscheidungstabellen sollte daher interaktiv erfolgen.[1)] Im folgenden ist der *interaktive* Ablauf einer rechnergestützten Erstellung von Entscheidungstabellen modellhaft zusammengestellt.

---

1) Zur interaktiven Erstellung von Entscheidungstabellen, allerdings nur begrenzter Entscheidungstabellen, vgl. Vieweg (Konstruktion von Entscheidungstabellen), S. 23 ff.

Teilschritt der	Manuell auszuführen	Maschinell ausführbar
Definitionen von Bedinngungen, Bedingungsanzeigern, Aktionen und Aktionsanzeigern als elementare syntaktische	Vollständige Eingabe. Bedingungsanzeiger sollten logisch disjunkt sein.	
Bildung der Basisfolge von Bedingungen und Aktionen.	Logische Abhängigkeiten bei Bedingungen und Aktionen, die bei der Bildung der Basisfolgen berücksichtigt werden müssen, sind einzugeben.	Reihenfolgeforderungen können als Präzedenzmatrix gespeichert werden. Jeweils eine zulässige Basisfolge der Bedingungen und Aktionen kann maschinell ermittelt werden.
Definition logischer Abhängigkeiten zwischen den Bedingungen, die die Regelentwicklung beeinflussen können.	Eingabe logischer Abhängigkeiten in Form Boolescher Ausdrücke[1)] oder gekennzeichneter Entscheidungsregeln, bzw. -teilregeln [2)] oder Matrizen [3)]	Speicherung der logischen Abhängigkeiten in Matrixform. Bei Eingabe der Abhängigkeiten in Matrixform kann das Grundmuster der Matrix automatisch aufbereitet werden.
Definition logischer Abhängigkeiten zwischen den Aktionen, die zur gemeinsamen Ausführung bestimmter Aktionsgruppen führen.	Eingabe in Form Boolescher Ausdrücke, gekennzeichneter Regeln bzw. Teilregeln oder Matrizen.	Speicherung und ggf. Aufbereitung des Grundmusters der Matrix.
Definition logischer Beziehungen zwischen Bedingungen und Aktionen.	Eingabe in Form Boolescher Ausdrücke, von Regeln bzw. Teilregeln oder in Form der Entscheidungsmatrix.	Bei Arbeiten mit der Entscheidungsmatrix Aufbereitung einer Matrix aus den Bedingungen mit allen Anzeigern

---

1) Vgl. Verhelst (Constructing decision tables), S. 31 f.

2) Beispielsweise kann das in Abschnitt 1.2.2.2.3 erläuterte Ausrufezeichen einen Bedingungsanzeiger als einzig möglichen im Rahmen der definierten Bedingungskonstellation ausweisen. Der Ausschlußanzeiger kann benutzt werden, um undefinierte Bedingungen im Rahmen einer bestimmten Bedingungskonstellation zu kennzeichnen.

3) Aus allen Bedingungen wird eine symmetrische Matrix erstellt und es werden Abhängigkeiten als Matrixelemente eingetragen.

4) Falls mit der Entscheidungsmatrix gearbeitet wird und die Eingabe am Bildschirm erfolgt, empfiehlt sich eine Matrixform, bei der jedem Bedingungsanzeiger eine separate Zeile und jedem Aktionsanzeiger eine separate Spalte zugeordnet wird. Die Kennzeichnung logischer Abhängigkeiten kann dann durch "1" oder ein anderes Zeichen im Schnittpunkt von Bedingungsanzeigerzeile und Aktionsanzeigerspalte erfolgen.

Teilschritt der Tabellenerstellung	Manuell auszuführen	Maschinell ausführbar
		als Zeilenelementen und den Aktionen mit allen Anzeigern als Spalten - elementen. Jede manuell eingegebene Beziehung zwischen Bedingung und Aktion wird automatisch daraufhin geprüft, ob sie nicht zu den vorher definierten logischen Abhängigkeiten im Widerspruch steht. Nach manueller Kennzeichnung aller logischen Beziehungen in der Matrix kann die Matrix unter Anwendung des erweiterten Quine-McCluskey-Verfahrens minimiert werden.[1)]
Erstellung des ersten Tabellenentwurfs im Format der Eintreffer-Entscheidungstabelle		Die im vorhergehenden Schritt erstellte Entscheidungsmatrix läßt sich unter Beachtung der sonstigen definierten logischen Abhängigkeiten automatisch in eine Eintreffer-Entscheidungstabelle umwandeln[2)] und minimieren. Bedingungskonstellationen, die zwar logisch zulässig sind, denen jedoch in der Entscheidungsmatrix keine Aktion zugewiesen wurde, werden besonders gekennzeichnet.
Vervollständigung des Tabellenentwurfs.	Ergänzung von Aktionsanzeigern. Möglicherweise auch Ergänzung zusätzlicher Bedingungen und Aktionen.	Jedes zusätzlich eingegebene Tabellenelement wird daraufhin überprüft, ob es mit den vorher definierten logischen Beziehungen und Abhängigkeiten vereinbar ist.

1) Vgl. Abschnitt 2.1.2.1.2.2
2) Zum Verfahren der Formattransformation vgl. Abschnitt 2.1.2.3

Die Schwäche des dargestellten interaktiven Verfahrens liegt in seiner Begrenzung auf die Erstellung von Einzeltabellen aus relativ kleinen Bedingungs-/Aktionsgruppen. Für die Praxis dürfte eine interaktive Erstellung von Entscheidungstabellen erst dann Bedeutung erlangen, wenn die Rechnerunterstützung auch das Vorfeld der Definition und Strukturierung des Entscheidungsfeldes abdeckt.

5. Entscheidungstabellen als Hilfsmittel der Systemimplementierung

Ebenso wie der Systementwurf lässt sich auch die Systemimplementierung in eine organisatorische und eine systemtechnische Teilphase zerlegen. Die Implementierung setzt einen abgeschlossenen organisatorischen und systemtechnischen Entwurf voraus. Da der Einsatz von Entscheidungstabellen nur die Gestaltung der *Verarbeitungsprozesse* eines Informationssystems beeinflusst, wird der Schwerpunkt der Darstellungen auf die Implementierung dieser Elementkategorie gelegt.

5.1 Entscheidungstabellen als Hilfsmittel der systemtechnischen Implementierung

Nach einer Erörterung der Teilaufgaben der systemtechnischen Implementierung von Informationsverarbeitungsprozessen werden Ansätze einer methodischen Programmierung und die Möglichkeiten der Einbeziehung der Entscheidungstabellentechnik diskutiert. Der Einfluss der Entscheidungstabellen-Anwendung auf die Programmübersetzung und -verifikation sowie auf die Programmoptimierung ist Gegenstand der darauf folgenden Abschnitte.

5.1.1 Teilaufgaben der systemtechnischen Implementierung von Informationsverarbeitungsprozessen

Unter der Voraussetzung, dass die benötigten Algorithmen während des *systemtechnischen Entwurfs* bereits konzipiert wurden, besteht die erste Teilaufgabe der *systemtechnischen Implementierung* darin, die Algorithmen so zu detaillieren, dass die Codierung in einer Programmiersprache möglich wird. Für häufig vorkommende Datenverarbeitungsprobleme bzw. -problemteile können Standardalgorithmen[1)] unverändert oder modifiziert benutzt werden. Individuell zu entwickelnde Algorithmen müssen bis zu einer Ebene konkretisiert werden, in der

---

1) Solche Standardalgorithmen finden sich beispielsweise in Knuth (Computer programming) und in jeder Ausgabe der Communications of the ACM.
Für kommerzielle Datenverarbeitungsanwendungen wird derzeit sogar versucht, eine Norm zu entwickeln, die einen allgemeingültigen Algorithmus für die Verarbeitung von Dateien nach Satzgruppen enthält. Vgl. DNA (Norm-Entwurf DIN 66220)

die einzelnen Teilschritte des Algorithmus in Sprachelemente der ausgewählten Programmiersprache umgesetzt werden können. Nach diesem Umsetzungsprozess - der Codierung - folgt die Übersetzung des Programms in die Maschinensprache. Die Verifizierung des Programms schließt sich an. Sie hat nachzuweisen, dass die ursprünglich durch den Entwurf spezifizierte Informationsverarbeitungsaufgabe durch das Programm erfüllt wird.

Ein verifiziertes Programm kann abschließend optimiert werden. Durch die Optimierung werden Leistungsmerkmale des Programms, wie beispielsweise Laufzeit und Speicherbedarf, verändert. Eine Änderung des Sachziels des Informationsverarbeitungsprozesses erfolgt jedoch nicht.

5.1.2 Methodische Programmierung unter Einsatz von Entscheidungsstabellen

Unter allen Ansätzen einer methodischen Programmierung, die derzeit praktiziert oder diskutiert werden, nehmen die Modulare Programmierung und die Strukturierte Programmierung eine herausragende Stellung ein. Ihr Leistungsumfang soll dargestellt und mit dem Leistungsumfang der Entscheidungstabellentechnik verglichen werden. Abschließend soll der Versuch einer Methodenintegration unternommen werden.

5.1.2.1 Modulare Programmierung

Unter den Begriffen "Unterprogrammtechnik" und "Overlay -Technik" wurden bereits während der Anwendungszeit der ersten beiden Rechnergenerationen bestimmte Formen der Strukturierung größerer Programme praktiziert, die als Vorläufer der Modularen Programmierung gelten können. Während die Programmierer zu dieser Zeit jedoch häufig aus Gründen der Hauptspeicherbegrenzung zu einer Aufteilung ihrer Programme gezwungen waren, stehen bei den heutigen Anwendern der Modularen Programmierung andere Motive im Vordergrund. Ein besserer Programmentwurf, einfacheres Testen, höhere Zuverlässigkeit des Programmsystems, erleichterte Programmwartung, höhere Programmiererproduktivität und wirksamere Projektsteuerungs- und – überwachungsmöglichkeiten stehen an der Spitze der Erfahrungen, die mit Modularer Programmierung gemacht wurden.[1)] Nach einer Definition der wichtigsten Merkmale der Modularen Programmierung und einer Darstellung

1) Vgl. o V. (Using modular programming), S. 3

neuerer Erkenntnisse über die Prinzipien eines modularen Entwurfs soll erläutert werden, wie die Entscheidungstabellentechnik in das Konzept der Modularen Programmierung passt.

5.1.2.1.1 Definition der Modularen Programmierung

Die Modulare Programmierung ist eine Technik, Programme in Form untereinander verknüpfter kleinerer Einheiten (Moduln) zu gestalten, deren Ausführung nach Standardregeln gesteuert wird.[1)]

Häufig wird in der Literatur gefordert, dass jeder Modul getrennt von den andern erstellt und getestet werden kann.[2)] Der mit der Ausführung solcher Moduln verbundene Aufwand ist bei der heutigen Hardware- und Systemsoftwaretechnik jedoch noch so hoch, dass Optimierungsüberlegungen dazu geführt haben, zusätzlich zu den "externen" Moduln auch "interne" Moduln zuzulassen. So könnte beispielsweise ein COBOL -Programm so modularisiert werden, dass jede SECTION als interner Modul gilt. Ein mit Hilfe von CALL aufgerufener Modul wäre ein externer Modul.[3)] Von jedem Modul, ob durch interne oder externe Modularisierung entstanden, wird gefordert, dass er nur einen Eingang und einen Ausgang hat.[4)] Der Aufruf eines Moduls erfolgt dann immer in der gleichen Weise. Die Aufrufanweisungen können jedoch für interne und externe Moduln unterschiedlich sein.

Ein wesentliches Kennzeichen der Modularen Programmierung ist, dass die Größe eines Moduls limitiert ist. Er soll nur eine einzige logische Funktion (oder Teilaufgabe) beinhalten.[5)] Da der Begriff der logischen Funktion nicht immer klar definierbar und daher auch nicht operational ist, wird versucht, über die Begrenzung des Umfangs eines Moduls die Trennung der Funktionen zu erzwingen. In Theorie und Praxis der Modularen Programmierung wird jedoch kein einheitlicher Maßstab zur Definition des wünschenswerten Umfanges eines Moduls ver-

---

1) Vgl. dazu Maynard (Modular programming), S. 6 und o. V. (Using modular programming), S. 10
2) Vgl. Maynard (Modular programming), S. 6 f.; Cohen (Defining the module), S. 34; Lee (Modulare Programmierung), S. 531; Liskov (Reliable software systems), S. 192
3) Vgl. o. V. (Using modular programming), S. 32 f. und Armstrong (Modular Programming), S. 73 ff. und S. 101 ff.
4) Vgl. o. V. (Using modular programming), S. 22 f.; Cohen (Defining the Module), S. 34
5) Vgl. Cohen (Defining the module), S. 34; Maynard (Modular programming), S. 6 f.

wendet. Einige Autoren empfehlen, die Anzahl der Anweisungen, die ein Modul enthält, als Maßstab zu benutzen. Beispielsweise schlägt Myers vor, jeden Modul, der weniger als 5 und mehr als 100 Quellprogramm-Anweisungen enthält, einer besonderen Analyse zu unterziehen. Moduln mit weniger als 5 Anweisungen können kaum eine selbständige Funktion beinhalten, und bei Moduln mit über 100 Anweisungen besteht der Verdacht, dass sie mehrere Funktionen abdecken.[1)]

In der Praxis der Modularen Programmierung werden als weitere Maßstäbe verwendet:

- die Anzahl von Bedingungen,
- die Anzahl maximal möglicher Wege der Programmausführung,
- der Umfang des Programmablaufplans (in der Regel ein DIN-A4-Blatt oder ein vergleichbares Maß),
- die Zeit, die zur Entwicklung und zum Test des Moduls erforderlich ist.[2)]

5.1.2.1.2 Gestaltungsprinzipien der Modularen Programmierung

Aus den dargestellten Kriterien zur Beschränkung der Modulgröße wird das Bemühen erkennbar, die *Komplexität* eines Moduls in Grenzen zu halten. Das ist das Hauptanliegen der Modularen Programmierung.

Allgemein wird jedoch erkannt, dass eine mehr oder weniger mechanische Begrenzung der Modulgröße das Problem nicht löst, sondern bei ungünstiger Aufteilung der Verarbeitungsprozesse auf die Moduln die Komplexität eines Programmsystems in der Komplexität der Beziehungen der Moduln untereinander ihren Ausdruck findet. Die Forderung zur Minimierung der Beziehungen der Moduln untereinander gilt deshalb als Grundprinzip eines modularen Entwurfs.[3)] Operational ist diese Forderung jedoch nur dann, wenn genau definiert werden kann, was als "Beziehung" von Moduln gilt. Vielfach wird die Auffassung vertreten, dass eine Beziehung zwischen zwei Moduln dann besteht, wenn ein Modul einen anderen Modul aufruft. Myers hat eine ausführliche Arbeit vorgelegt, in der die möglichen Beziehungen zwischen zwei Moduln sorg-

1) Myers (Composite design), S. 34 f.
2) Vgl. o. V. (Using modular programming), S. 54
3) Vgl. dazu Myers (Composite design), S. 14

fältig analysiert werden.[1)] Er unterscheidet fünf Stufen in einer Skala geringer bis hochgradiger Verknüpfung:

1. Verknüpfung über Ein-/Ausgabedaten. Von einem zum anderen Modul werden nur Datenelemente übergeben.
2. Gemeinsame Benutzung von Datenelementen. Beide Moduln benutzen und verändern dieselben Variablen.
3. Ein Modul beeinflusst den anderen Modul durch Übergabe von Daten, die die Programmsteuerung des anderen Moduls modifizieren. Als Beispiel kann das Setzen von Programmschaltern, die vom anderen Modul benutzt werden, gelten.
4. Ein Modul benutzt ein mit dem EXTERNAL -Attribut versehenes Datenelement,[2)] das im anderen Modul definiert ist und dort ebenfalls benutzt wird.
5. Zwei Moduln sind durch den Programminhalt miteinander verknüpft.
   Beispielsweise benutzt ein Modul einen Teil der Anweisungen des anderen Moduls.

Auf den wertvollen Erkenntnissen von Myers aufbauend, ist es möglich, drei elementare Modulverknüpfungsarten abzuleiten, die prozedurale Verknüpfung, die dateninduzierte Verknüpfung und die programminduzierte Verknüpfung.

Zwei Moduln sind *prozedural* verknüpft, wenn die Programmsteuerung vom ersten auf den zweiten Modul übergeht. Der zweite Modul führt die Programmsteuerung von seinem Eingang aus weiter.

Die prozedurale Verknüpfung von Moduln kann drei Erscheinungsformen haben. Bei sequentieller Verknüpfung geht die Programmsteuerung vom Ausgang des ersten Moduls auf den Eingang des zweiten Moduls über. Bei iterativer Verknüpfung liegen die Moduln in einem Zyklus. Nach Ausführung des zweiten Moduls geht die Steuerung wieder zum Eingang des ersten Moduls zurück. Bei verschachtelter Verknüpfung kann aus

---

1) Vgl. dazu Myers (Composite design), S. 22 ff. Myers verweist mehrfach darauf, dass ein Teil seiner Ideen Larry Constantine zuzuschreiben ist, dessen Buch "Fundamentals of Program Systems Design" in Vorbereitung sei.

2) Die explizite Kennzeichnung eines Datenelementes mit einem EXTERNAL- Attribut ist in PL/I möglich. Ähnliche Definitionsmöglichkeiten bieten auch Assemblersprachen.

einer beliebigen Position des ersten Moduls heraus der zweite Modul aufgerufen werden. Nach vollständiger Bearbeitung des zweiten Moduls geht die Steuerung wieder an den ersten Modul zurück, und dieser fährt mit der Ausführung der auf die Aufrufanweisung folgenden Anweisung fort.

Die *dateninduzierte* Verknüpfung von Moduln hat zwei Erscheinungsformen. Im ersten Fall übergibt der erste Modul an den zweiten Modul Datenelemente, die dieser weiter bearbeitet. Sowohl im ersten als auch im zweiten Modul haben die Datenelemente den Charakter lokaler Variablen, d. h. sie müssen auf unterschiedlichen Speicherplätzen gespeichert sein. Im zweiten Fall benutzen und verändern beide Moduln Datenelemente, die ihren Speicherplatz mit Übergabe der Programmsteuerung an den zweiten Modul nicht verändern. Sie haben den Charakter globaler Variablen.[1)]

Die *programminduzierte* Verknüpfung von Moduln hat ebenfalls zwei Erscheinungsformen. Im ersten Fall greift der erste Modul - beispielsweise durch Verändern der Stellung einer Programmweiche im zweiten Modul - aktiv in die Programmlogik des zweiten Moduls ein. Dieser Fall soll als "Programmbeeinflussung" gekennzeichnet werden. Im zweiten Fall übernimmt der erste Modul einen Teil der Prozeduranweisungen des zweiten Moduls, so dass beide Moduln gemeinsame Programmteile aufweisen.

Da die erläuterten elementaren Modulverknüpfungsarten kombiniert auftreten können, aber nicht notwendigerweise auch auftreten müssen, entspricht die Beziehung zweier Moduln einem der in der folgenden Tabelle definierten 36 Fälle.

---

1) Wie bereits erläutert, unterscheidet Myers bei den Modulverknüpfungsarten die Verbindung durch Datenelemente mit EXTERNAL -Attribut und die Verbindung durch Datenelemente, die in einem allen Moduln gemeinsamen Datenbereich (z. B. COMMOM-Bereich bei FORTRAN) gespeichert sind. Dieser Unterscheidung kann hier nicht gefolgt werden. In beiden Fällen handelt es sich um globale Variablen, die ihren Speicherplatz nicht verlassen und von mehreren Moduln aus verändert werden. Positive und negative Auswirkungen dieser Modulverknüpfungsart sind in beiden Fällen gleich.

Daten-induzierte Verknüpfungen		keine	keine	keine	keine	lok. Var.	lok. Var.	lok. Var.	lok. Var.	glob. Var.	glob. Var.	glob. Var.	glob. Var.
Prozedurale Verknüpfungen		keine	seq.	it.	ver-schach-telt	keine	seq.	it.	ver-schach-telt	keine	seq.	it.	ver-schach-telt
Programm-induzierte Verknüp-fungen	keine	1	2	3	4	5	6	7	8	9	10	11	12
	Be-einfl.	13	14	15	16	17	18	19	20	21	22	23	24
	Gem. Proz.	25	26	27	28	29	30	31	32	33	34	35	36

Die Forderung, die Beziehungen zwischen den unterschiedlichen Moduln eines Programmsystems zu minimieren, kann nun operationalisiert werden.

Alle *programminduzierten* Beziehungen sind strikt zu verbieten. Sie stehen im Gegensatz zu der Forderung, dass jeder Modul nur einen Eingang und einen Ausgang haben darf, und verringern Zuverlässigkeit und Wartungsfreundlichkeit der Programme.

Nur noch 12 unterschiedliche Fälle von Modulverknüpfungen verbleiben nach Eliminierung der programminduzierten Beziehungen. Davon beinhalten die Fälle 9 - 12 die Benutzung globaler Variablen. Da auch globale Variablen die Wartungsfreundlichkeit und Flexibilität eines Programmsystems einschränken, sollte ihre Verwendung auf Fälle beschränkt bleiben, in denen der Optimierung des Programmsystems höhere Priorität eingeräumt werden muss als seiner Wartungsfreundlichkeit und Flexibilität. Globale Variablen ersparen Daten- und Adressübertragungen. Andererseits kann sich die Änderung eines Moduls, die auch auf eine globale Variable Einfluss nimmt, auf alle anderen Moduln auswirken, die ebenfalls diese Variable benutzen, und so die Zuverlässigkeit des Programmsystems vermindern.

Die verbleibenden 8 Fälle können als unproblematisch eingestuft werden. Sie beeinträchtigen weder die Wartungsfreundlichkeit noch die Flexibilität und Zuverlässigkeit des Programmsystems.

5.1.2.1.3 Entscheidungstabellen im Konzept der Modularen Programmierung

Die Anwendung der Modularen Programmierung führt zu einer Unterteilung eines Programmsystems in kleinere Programmeinheiten, die jeweils eine

logische Funktion abdecken und deren Beziehungen untereinander minimiert wurden. Diese Art der methodischen Programmierung steht nicht im Gegensatz zur Anwendung von Entscheidungstabellen bei der Programmerstellung. Vielmehr lässt sich aufzeigen, dass aus einer Kombination der Modularen Programmierung mit der Entscheidungstabellentechnik beide Methoden Vorteile ziehen.

Die von der Modularen Programmierung geforderte Zuweisung von nur einer Funktion zu jedem Modul lässt sich in der Praxis nicht immer auf Anhieb realisieren. Es können Zweifel darüber bestehen, in welcher Weise eine umfassende Funktion sinnvoll weiter unterteilt werden soll. Hier kann die konsequente Anwendung der Entscheidungstabellentechnik die Lösung bringen. Auf eine Aufgliederung der Funktion wird verzichtet und sofort zu einer Analyse von Bedingungen, Aktionen und ihren Beziehungen übergegangen. Ein Entscheidungstabellenverbund wird erstellt, und erst auf der Basis dieses Entscheidungstabellenverbunds wird modularisiert.[1)] Da jede Entscheidungstabelle logisch verknüpfte Bedingungen und Aktionen auf sich konzentriert, trägt sie vieles zur Entwicklung der Teilfunktionen bei.[2)] Die intensive Anwendung von Entscheidungstabellen in Verbindung mit einem Entscheidungstabellenvorübersetzer erlaubt es auch, größere Moduln zuzulassen, ohne Nachteile bei der Transparenz des Programms in Kauf nehmen zu müssen.

Zur Bestimmung der zulässigen Modulgröße werden nur diejenigen Anweisungen in die Berechnung einbezogen, die die Bedingungen und Aktionen der Entscheidungstabelle bilden. Die Anzahl der aus der Vorübersetzung resultierenden Quellprogrammanweisungen ist in diesem Zusammenhang ohne Bedeutung.

Schließlich sei darauf hingewiesen, dass sich die Entscheidungstabelle sehr gut zur Implementierung von Steuermoduln, die häufig auch die Ein-/Ausgabe-Operationen enthalten, eignet. Ein Steuermodul enthält alle Aufrufe der Verarbeitungsmoduln, Plausibilitätsprüfungen und die Ein-/Ausgabe-Fehlerbehandlung.[3)] Die Häufung von Bedingungsprüfungen ist ein charakteristisches Merkmal solcher Steuermoduln, deren Implementierung durch die Anwendung von Entscheidungstabellen erheblich erleichtert wird.

---

1) In Abschnitt 4.3.2.3 wurde erläutert, wie das mit Hilfe graphentheoretischer Verfahren geschehen kann.

2) Auf die nützliche Kombination von Entscheidungstabellentechnik und Modularer Programmierung wies auch Hicks in einem Interview hin. Vgl. Canning (Modular COBOL programming), S. 6

3) Vgl. Lee (Modulare Programmierung), S. 534

Auch die Entscheidungstabellentechnik kann aus einer Kombination mit der Modularen Programmierung Nutzen ziehen. Große, auf Entscheidungsstabellen basierende Programmsysteme können bei einer monolithischen Implementierungstechnik leicht mehrere hundert oder sogar tausend Entscheidungstabellen umfassen. Trotz Anwendung von Entscheidungsstabellen kann dann die Transparenz des Systems verlorengehen. Die Modularisierung des monolithischen Entscheidungstabellenverbunds in Moduln, die nicht mehr als 10 - 20 Entscheidungstabellen enthalten, stellt die Transparenz wieder her.

5.1.2.2 Strukturierte Programmierung

Die Auffassung über den Inhalt der *Strukturierten Programmierung*, die ihre Bezeichnung offensichtlich von Dijkstra erhalten hat, [1] sind nicht einheitlich. In einem ersten Unterabschnitt sollen deshalb zunächst die Merkmale der Strukturierten Programmierung diskutiert werden. Vorschläge einer Einordnung der Entscheidungstabelle in das Konzept der Strukturierten Programmierung folgen.

5.1.2.2.1 Merkmale der Strukturierten Programmierung

In den Veröffentlichungen zur Strukturierten Programmierung herrscht Einigkeit darüber, dass jedes Programm aus wenigen elementaren Ablaufstrukturen zusammengesetzt werden muss, wobei diese Ablaufstrukturen hintereinander gereiht oder ineinander verschachtelt werden können.

In manchen Veröffentlichungen wird die Strukturierte Programmierung auch mit einer "GO TO" -losen Programmierung gleichgesetzt. Schließlich findet sich häufig auch die Forderung, ein System in eine Hierarchie von Abstraktionsebenen zu zerlegen und bei der Realisierung die Technik der Top - down -Programmierung anzuwenden.

5.1.2.2.1.1 Elementare Ablaufstrukturen

Jacopini[2] hat das Theorem aufgestellt und bewiesen,[3] dass jedes Programm aus den elementaren Ablaufstrukturen Sequenz, Entscheidung und Iteration aufgebaut werden kann.

---

1) Diese Ansicht äußert Knuth und verweist auf Dijkstra (Structured programming). Vgl. Knuth (Structured programming), S. 291
2) Vgl. Bohm; Jacopini (Flow diagrams), S. 366 ff.
3) Vgl. dazu auch den ausführlicheren Beweis bei Mills (Structured programming), S. 54 ff.

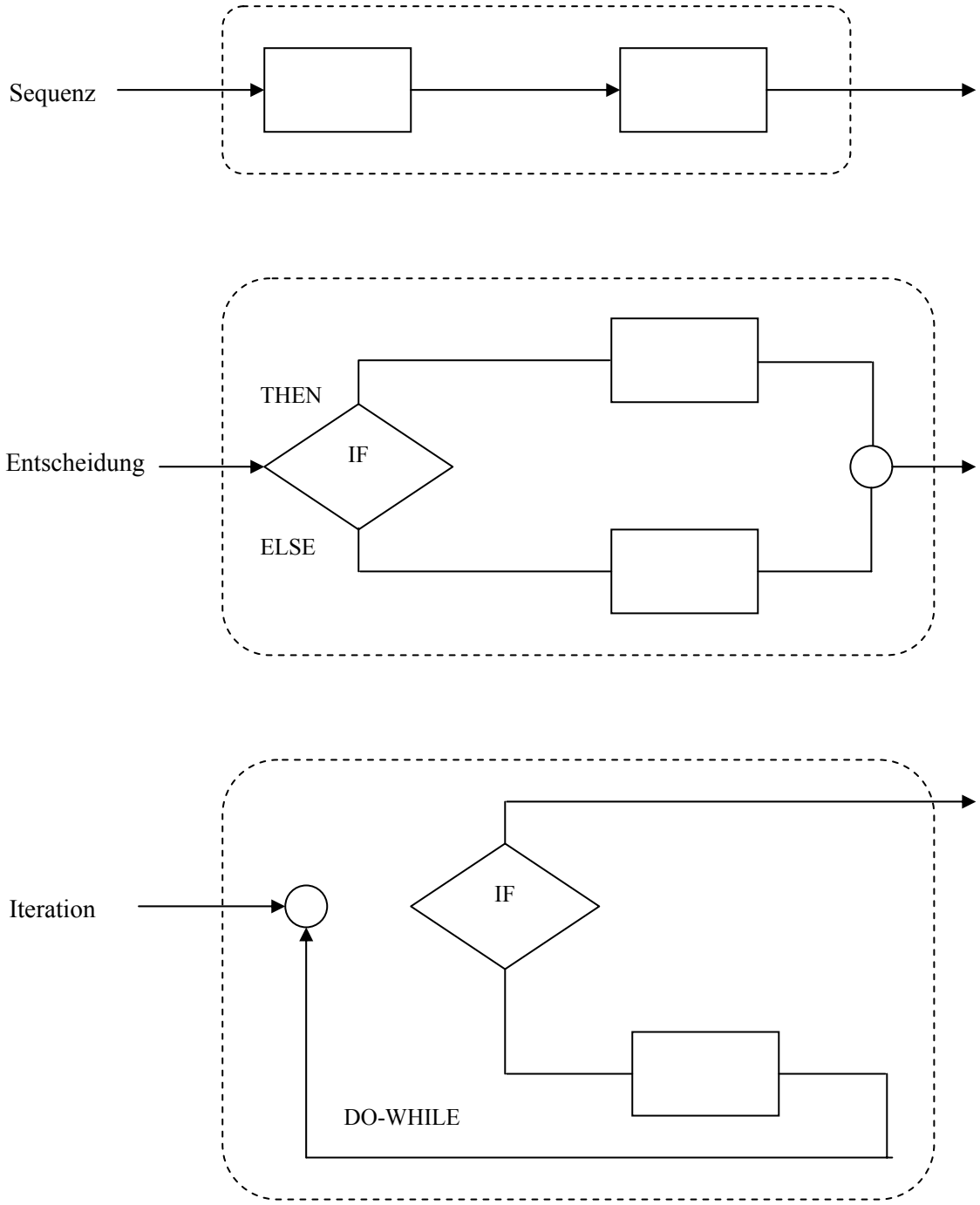


Abb. 9: Elementare Ablaufstrukturen

Diese Ablaufstrukturen sind "elementar", weil sie nicht mehr in andere Ablaufstrukturen zerlegbar sind.

Als Alternative zur DO –WHILE -Form der Iteration entsteht die DO –UNTIL -Ablaufstruktur, wenn zuerst die Aktion ausgeführt wird und danach die Bedingungsprüfung erfolgt:

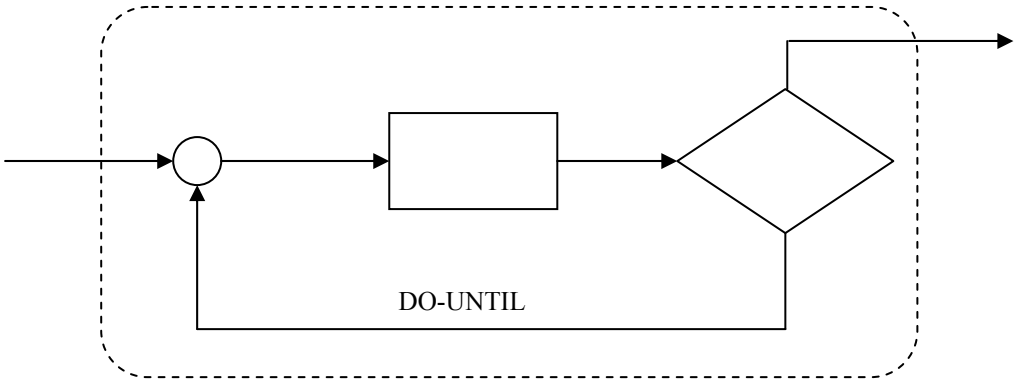


Abb. 10: Interaction mit DO-UNTIL

Die elementaren Ablaufstrukturen haben einige bemerkenswerte Eigenschaften. Sie haben jeweils einen Eingang und einen Ausgang und können demnach aus einer abstrakteren Betrachtung heraus als eine einzige Aktion angesehen werden. [1)] Außerdem kann ihre Verifikation nach bekannten Methoden erfolgen. Sequenz und Entscheidung lassen sich durch Enumeration, die Iteration durch mathematische Induktion verifizieren.[2)] Als weitere Ablaufstruktur kann die CASE -Anweisung als Fallunterscheidung ergänzt werden, um Bedingungen mit mehr als zwei Zuständen günstiger darstellen zu können:[3)]

---

1) Vgl. Dijkstra (Notes on structured programming), S. 19
2) Vgl. Dijkstra (Notes on structured programming), S. 19 f.; Mills (New math), S. 47
3) Die CASE -Anweisung geht auf einen Vorschlag von Hoare zurück.
Vgl. dazu Dijkstra (Notes on structured programming), S. 18

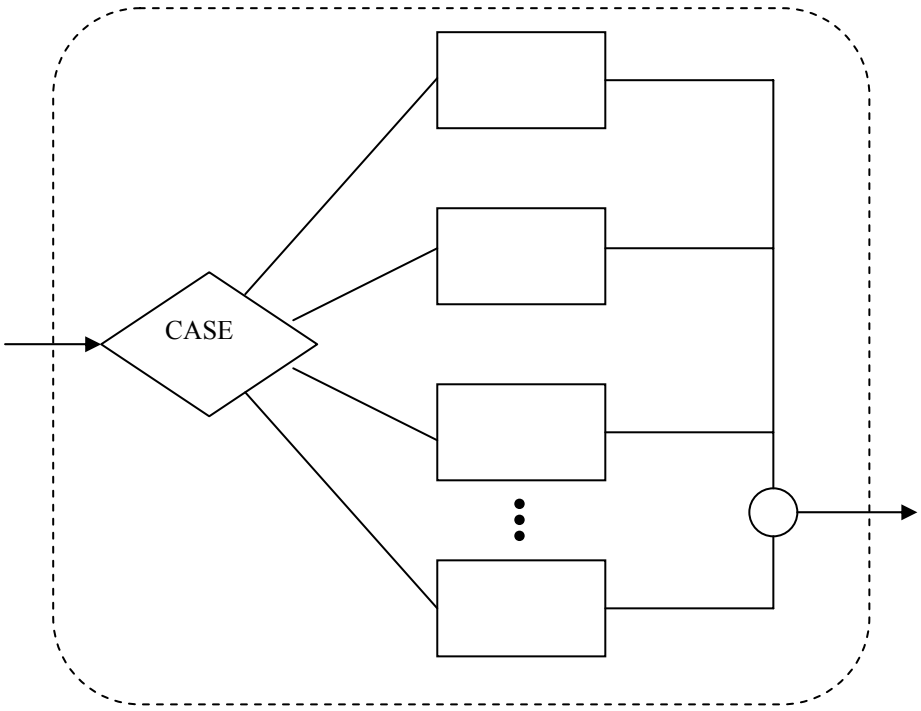


Abb. 11: Fallunterscheidung

Durch Aneinanderreihung und Ineinanderschachtelung dieser Ablaufstrukturen lässt sich nun jedes Programm entwickeln, vorausgesetzt, die verwendete Programmiersprache lässt die Bildung der benötigten Konstruktionen zu.[1)]

5.1.2.2.1.2 "GO TO" -lose Programmierung

In seinem mittlerweile berühmten Leserbrief an die Communications of the ACM[2)] teilt Dijkstra seine Beobachtung mit, dass die Qualität von Programmierern eine abnehmende Funktion der Häufigkeit von "GO TO" -Anweisungen in den von ihnen erstellten Programmen sei. Die Verwendung der "GO TO" -Anweisung erschwere die Interpretation des dynami-

1) Die derzeit überwiegend in der Praxis eingesetzten problemorientierten Programmiersprachen sind nicht unter Beachtung der Forderungen der Strukturierten Programmierung entwickelt worden. Dennoch lassen sich auf der Basis der definierten Sprachelemente die Forderungen der Strukturierten Programmierung in ihren wesentlichen Punkten erfüllen. Zur Strukturierten Programmierung mit COBOL vgl. Floyd (Strukturierte Programmierung), S. 39 ff.

2) Vgl. Dijkstra (GO TO statement), S. 147 f.

schen Verhaltens des Programms auf der Grundlage des statischen Programmtextes. Dijkstra stellt schließlich die Forderung auf, die "GO TO" -Anweisung aus höheren Programmiersprachen zu entfernen und jedes Programm aus wenigen elementaren Ablaufstrukturen zu konstruieren.[1] Dijkstra eröffnete damit die "GO TO" -Debatte, die in den folgenden Jahren stürmisch geführt wurde.[2]

Mit der kürzlich erfolgten ausführlichen Stellungnahme Knuths[3] wurde Klarheit darüber geschaffen, ob die "GO TO" -lose Programmierung Merkmal der Strukturierten Programmierung ist. Strukturierte Programme, so führt Knuth aus, entstehen nicht durch die Eliminierung von "GO TO" -Anweisungen aus Programmen, die auf irgendeine Weise entstanden sind. Vielmehr sollte sich als *Konsequenz* eines methodischen "strukturierten" Aufbaus von Programmen ergeben, dass das "GO TO" nur noch selten benutzt werden muß.[4] Knuth weist jedoch nach, dass es gewichtige Gründe für die Benutzung von "GO TO" -Anweisungen innerhalb strukturierter Programme geben kann. Beispielsweise werden häufig Programmzyklen in folgender Form aufgebaut:

```
prozedurname-l:
        programmtext-l
        IF B THEN GO TO prozedurname-2
        programmtext-2
        GO TO prozedurname-l

prozedurname-2:
```

"GO TO" -freie Konstruktionen von Programmzyklen aus den elementaren Ablaufstrukturen sind zwar möglich, jedoch aus Gründen geringer Effizienz unbefriedigend.[5] Ähnlich unbefriedigend ist es, anomale Beendigungen von Programmzyklen, zum Beispiel bei Eintreten von Fehlerbedingungen, ohne GO TO programmieren zu müssen. Ein mit Hilfe

---

1) Vgl. Dijkstra (GO TO statement), S. 147. Dijkstra erklärt sich darin beeinflusst von Peter Candin, Christopher Strachey, Heinz Zemanek, C. A. R. Hoare und Niklaus Wirth. Offensichtlich ist die Problematik der "GO TO" -Anweisung erheblich früher entdeckt worden. Dijkstra gebührt jedoch das Verdienst, für die nötige Publizität gesorgt zu haben. Vgl. dazu die historischen Anmerkungen bei Knuth (Structured programming), S. 264 ff.

2) Vgl. dazu Clark (GOTO -less programming); Hopkins (Case for the GOTO); Dickman (GOTO controversy); Leavenworth (Programming with (out) the GO TO); Wulf (Case against GOTO)

3) Vgl. Knuth (Structured programming), S. 261 ff.

4) Vgl. Knuth (Structured programming), S. 291; Vgl. auch Wirth (Well-structured programs), S. 257

5) Vgl. Knuth (Structured programming), S. 278 f.

von GO TO realisierter Fehlerausgang kann die Transparenz des Programms sogar erhöhen, weil damit ein anomales Ereignis im Programmablauf dokumentiert wird.[1)]

In vielen Fällen kann allgemein eine größere Effizienz eines Programms nur noch dann erzielt werden, wenn die wohldefinierten Strukturen wieder in weniger transparente, dafür aber laufzeit- oder speicherplatzgünstigere Anweisungsfolgen auch unter Verwendung von "GO TO" -Anweisungen transformiert werden. Eine solche Transformation kann nach mechanisch ausführbaren Regeln oder möglicherweise sogar automatisch erfolgen. Die laufende Programmwartung sollte jedoch nicht auf der Basis des so entstehenden Programms, sondern des ursprünglichen strukturierten Programms erfolgen.[2)]

Die Ideen Knuths lassen sich leicht in eine Forderung an die Entwickler von Kompilierern umsetzen: Warum sollte es nicht möglich sein, vor der mehr oder weniger schematischen Umsetzung der Quellsprache in die Zielsprache transparente aber aufwendige Strukturen in wirksamere Anweisungsfolgen aufzulösen?

5.1.2.2.1.3 Konzept der Abstraktionsebenen

Die Forderung, Programmsysteme stufenweise zu entwickeln und ein Programmsystem als Hierarchie mehrerer Abstraktionsebenen zu gestalten, kann als weiteres Merkmal der Strukturierten Programmierung gelten. Nur durch Abstraktion gelingt es, komplexe Programmsysteme in überschaubare Teilabschnitte zu zergliedern, ohne die Vollständigkeit dabei aufgeben zu müssen.[3)]

Das Prinzip der Abstraktion in der Programmierung als erster formuliert und seine konsequente Anwendung beschrieben zu haben, gebührt Dijkstra. Bei der Entwicklung eines Betriebssystems für Mehrprogrammbetrieb wurde das gesamte Programmsystem in fünf Abstraktionsebenen unterteilt, wobei die unterste Ebene auf die Hardwareeigenschaften unmittelbar Bezug nahm, die oberste Ebene dagegen auf die Benutzerbedürfnisse.[4)]

---

1) Vgl. Knuth (Structured programming), S. 293
2) Vgl. dazu Knuth (Structured programming), S. 294 f.
3) Vgl. Dijkstra (Humble programmer), S. 864; Hoare (Data structuring), S. 87; Wirth (Well-structured programs), S. 249
4) Vgl. Dijkstra ("THE"-multiprogramming system), S. 343

Die Gestaltung eines Systems nach dem Konzept der Abstraktionsebenen kann in der untersten Ebene damit beginnen, dass alle Funktionen beschrieben werden, die spezielle Hardwareeinrichtungen oder besondere Repräsentationen der Daten voraussetzen. In der nächst höheren Stufe können dann Funktionen beschrieben werden, in deren Umgebung bestimmte Hardware- und Datenbesonderheiten nicht mehr bekannt sind, diese Eigenschaften des Systems sind durch Abstraktion entfallen. Jede höhere Ebene der Abstraktion entsteht durch Reduzierung der Umgebung um weitere Eigenschaften.[1)]

Die beiden wichtigsten Prinzipien der Bildung von Abstraktionshierarchien sind:[2)]

1. Jede Abstraktionsebene kann über Ressourcen verfügen (z. B. Ein-/ Ausgabegeräte, spezielle Daten), zu denen andere Abstraktionsebenen keinen Zugriff haben;

2. Funktionen einer Abstraktionsebene dürfen keine Routinen höherer Abstraktionsebenen aufrufen, können jedoch Routinen niedrigerer Abstraktionsebenen benutzen.

Programmsysteme, die nach dem dargestellten Konzept der Abstraktionsebenen gestaltet sind, können als äußerst flexibel gelten. Bei einer notwendig werdenden Änderung einer Funktion bleiben alle Funktionen der höheren Abstraktionsebenen unbeeinflusst.[3)]

5.1.2.2.1.4 Top –down -Programmierung

Von einigen Autoren wird das Prinzip der Top –down -Programmierung als Merkmal der Strukturierten Programmierung bezeichnet.[4)] Darin sind zwei Komponenten enthalten: der Top -down -Entwurf und die Top –down -Implementierung.

Der *Top –down -Entwurf* eines Programmsystems schafft eine Hierarchie von Funktionen nach dem Konzept der Abstraktionsebenen, wobei der Entwurfsprozess mit der Definition der abstraktesten Ebene beginnt und sich schrittweise bis zur Definition der konkretesten Ebene bewegt. [5)]

---

1) Vgl. dazu Liskov (Reliable software systems), S. 193;
Dahl; Hoare (Hierarchical program structures), S. 209
2) Vgl. dazu Liskov (Reliable software systems), S. 193
3) Vgl. Liskov (Reliable software systems), S. 196 f.
4) Vgl. Mills (Structured programming), S. 53 f.; Miller; Lindamood (Top-down-approach), S. 55
5) Vgl. Mills (Top-down programming), S. 50 ff.; Mills (Structured programming), S. 52 f.

Das Konzept der *Top –down -Implementierung*[1)] fordert, die Moduln der höchsten (abstraktesten) Ebene zuerst zu codieren und zu testen. Um die Moduln der höchsten hierarchischen Ebene, bei denen es sich in der Regel um Steuermoduln handeln wird, überhaupt testen zu können, ohne Moduln der darunter liegenden Ebenen verfügbar zu haben, wird die Technik der Modulsubstitution angewendet. Der noch nicht vorhandene Modul wird entweder durch einen "dummy module" ersetzt, der nur aus dem Modulnamen und einer Rücksprung-Anweisung besteht, oder es wird ein in seinen Funktionen stark eingeschränkter Modul implementiert, der erst später durch den vollständigen Modul ersetzt wird.[2)] Mit Hilfe der Technik der Modulsubstitution kann jede hierarchische Ebene fortschreitend bis zur untersten Ebene, die keine Substitutionen mehr erfordert, implementiert werden.

Die Technik der Top –down -Implementierung ist nicht erst im Zusammenhang mit der Strukturierten Programmierung "erfunden" worden. Vielmehr handelt es sich dabei um ein in der Modularen Programmierung wohlbekanntes Verfahren.[3)] Auch die Idee, die einzelnen Moduln in ihrer Größe zu beschränken, wurde von der Modularen Programmierung in die Strukturierte Programmierung übernommen. Häufig findet sich der Hinweis, dass jeder Modul nicht größer als eine Seite mit etwa 50 Anweisungszeilen sein sollte.[4)]

5.1.2.2.2 Entscheidungstabellen im Konzept der Strukturierten Programmierung

Entscheidungstabellen lassen sich auf zweierlei Arten mit dem Konzept der Strukturierten Programmierung vereinbaren.

Zum einen kann die Entscheidungstabelle als höhere Struktur beim Aufbau eines Strukturierten Programms zugelassen werden. Damit würde eine Kurzform zur Darstellung von Entscheidungssituationen bereitstehen, die eine unübersichtliche Verschachtelung von Elementarstrukturen vermeiden würde. Es sei darauf hingewiesen, dass auch das

---

1) Vgl. dazu Mills (Chief programmer teams), S. 24 ff.
2) Vgl. Maynard (Modular programming), S. 65 f.
3) Vgl. Maynard (Modular programming), S. 65 f;
Canning (Modular COBOL programming), S. 6; o. V. (Using modular programming), S. 30 f.; Canning (Advent of structured programming), S. 6 f.
4) Vgl. Mills (Structured programming), S. 12; Miller; Lindamood (Top-down-approach), S. 56

CASE als Kurzform einer Struktur aus verschachtelten IF –THEN –ELSE -Anweisungen verstanden werden kann.[1)] Folgende Darstellungen sind demnach logisch äquivalent:

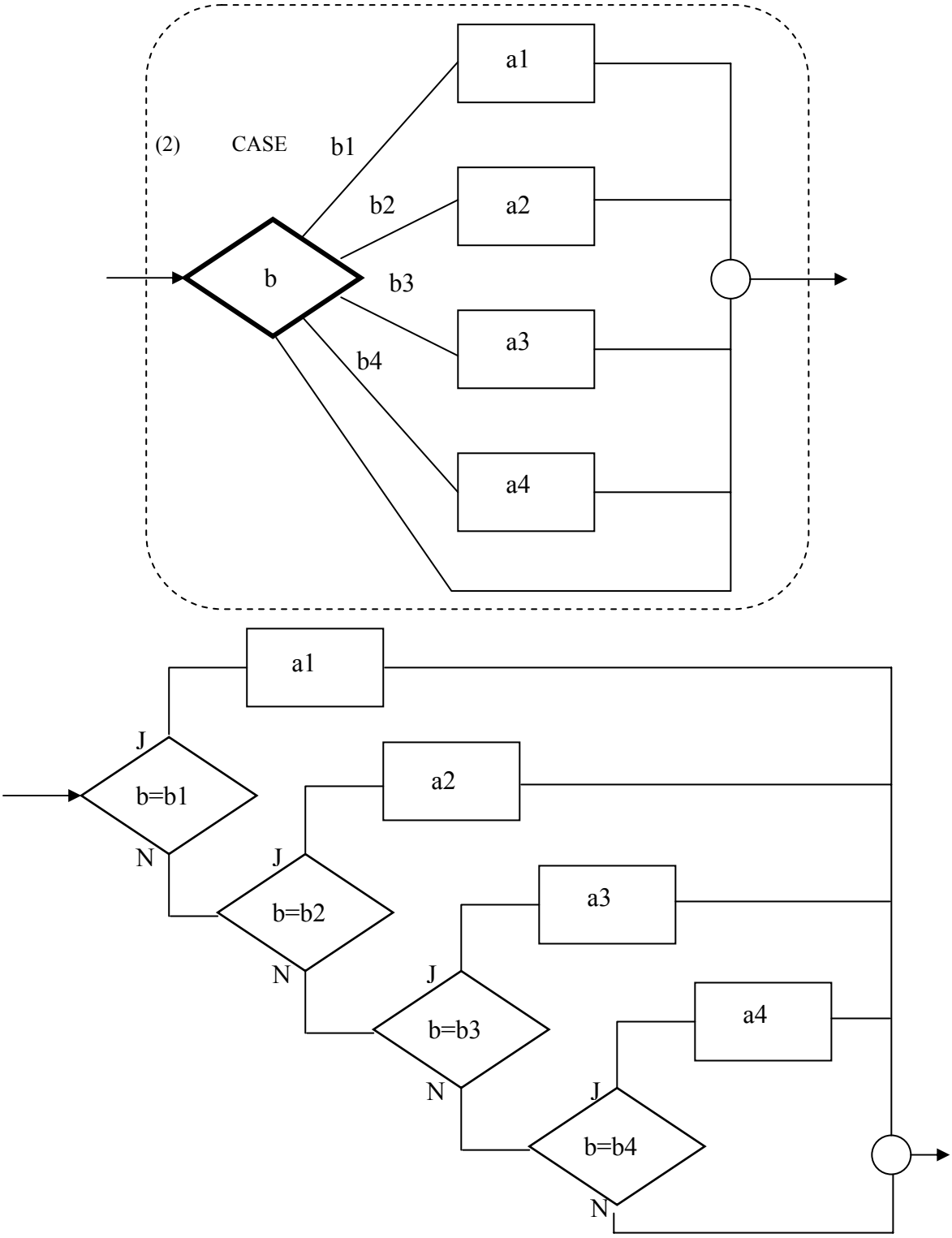


Abb. 12: Auflösung von CASE mit IF-THEN-ELSE

1) In diesem Sinn bezeichnet auch Mills das CASE als einen zusammengesetzten Baustein Strukturierter Programme. Vgl. Mills (Structured programming), S.42 f.

Eine höhere Struktur "DECISION" ist bereits von Mills vorgeschlagen worden.[1)]

DECISION				
PREDICATES				
p:	T		T	F
q:		F	T	T
r:		T		
s:	T	F	T	
FUNCTIONS				
f:		1	1	2
g:	1	1		
h:	3		2	3
i:	2			1
j:		2		2
k:		3	3	
LOOP	Y	Y	N	Y

Diese Entscheidungsstruktur enthält 4 Bedingungen, 6 Aktionen und einen Zyklus-Anzeiger LOOP, der eine Durchlaufwiederholung aufzeigt. T steht für TRUE, F für FALSE, Ziffern zeigen die Reihenfolge der Ausführung der Aktionen an. Damit hat Mills nichts anderes als eine Entscheidungsstabelle beschrieben. Entscheidungstabellen könnten somit ohne Gefahr einer Verwässerung des Konzepts der Strukturierten Programmierung neben den bekannten elementaren Strukturen angewendet werden. Zum anderen könnte jedoch versucht werden, die Erkenntnisse der Strukturierten Programmierung vollständig auf die Entscheidungstabellentechnik zu übertragen.

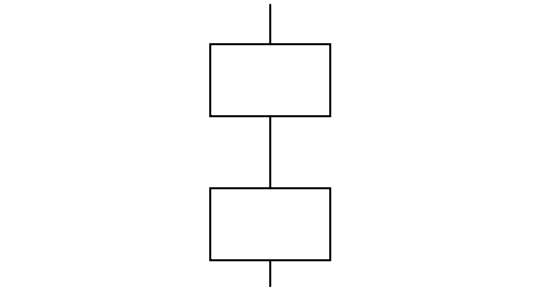
Dies erfordert erstens die Abbildung der elementaren Ablaufstrukturen als Entscheidungstabellen und zweitens eine der Verschachtelung der Elementarstrukturen adäquate Anweisung zum Aufbau komplexer Programmeinheiten aus den elementaren Strukturen.

---

1) Vgl. Mills (Structured programming), S. 46

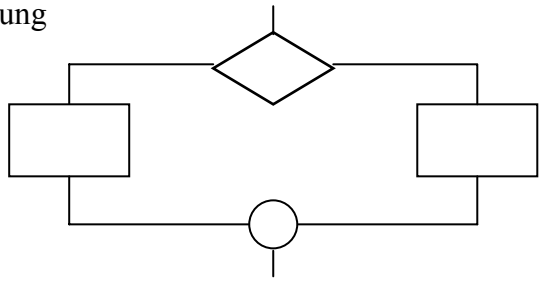
Die elementaren Ablaufstrukturen lassen sich ohne Schwierigkeiten in Entscheidungstabellen überführen.[1)]

Sequenz



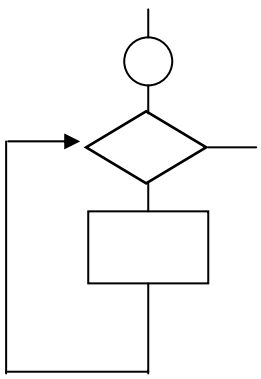
		R1
A1	Aktion α	X
A2	Aktion β	X
A3	AUSGANG	X

Entscheidung



		1	2
B1	Bedingung α	J	N
A1	Aktion γ	X	
A2	Aktion δ		X
A3	AUSGANG	X	X

Iteration



		1	2
B1	Bedingung b	J	N
A1	Aktion ε	X	
A2	Wiederholung	X	
A3	AUSGANG		X

Abb. 13: Elementare Entscheidungstabellen

1) Vgl. Strunz (Verträglichkeit der Strukturierten Programmierung) S. 19

Gegenüber den bisher erläuterten Definitionen der syntaktischen Einheiten der Entscheidungstabelle ergeben sich folgende Besonderheiten:

- Alle elementaren Entscheidungstabellen haben als letzte Aktion die Anweisung "AUSGANG".

- Zur Abbildung der Sequenz wird eine bedingungslose Tabelle benutzt. Bedingungslose- oder "Aktions-Tabellen" sind in der Entscheidungsstabellen-Literatur schon längst bekannt. Sie werden immer dann als notwendig. erachtet, wenn bei der Beschreibung eines Informationsverarbeitungsprozesses ausschließlich mit Entscheidungstabellen gearbeitet werden soll.[1)]

- Die elementaren Entscheidungstabellen zur Abbildung der Strukturen "Entscheidung" und "Iteration" beinhalten jeweils nur eine einfache Bedingung.

Neben den elementaren Ablaufstrukturen kann selbstverständlich auch die CASE -Anweisung als Entscheidungstabelle abgebildet werden.[2)]

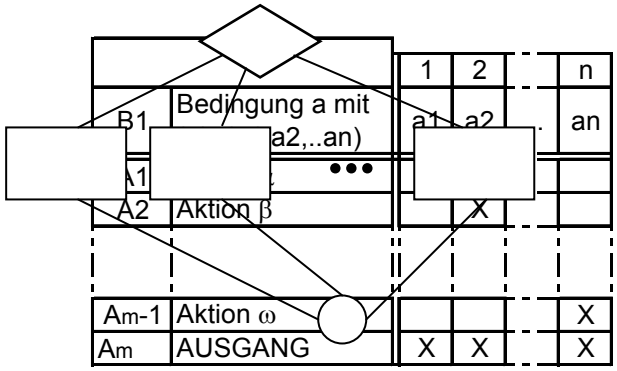


Abb. 14: CASE als Entscheidungstabelle

1) Vgl. SIGPLAN (DETAB/65), 5.16
2) Vgl. Strunz (Verträglichkeit der Strukturierten Programmierung), 5.20

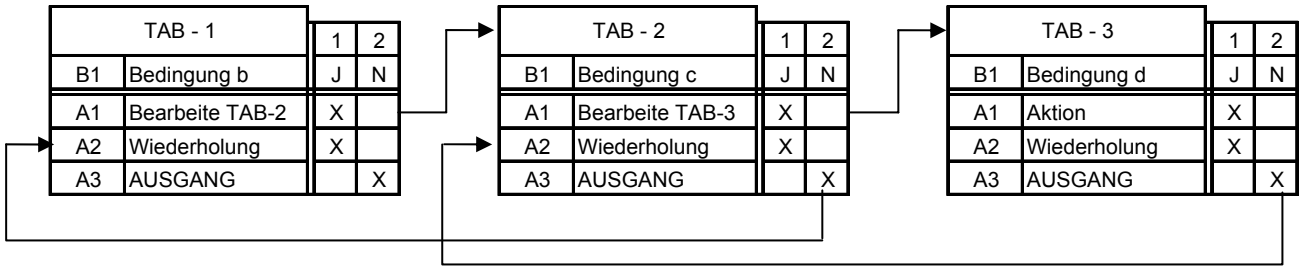
Analog zur Verschachtelung von elementaren Ablaufstrukturen lassen sich auch elementare Entscheidungstabellen verschachteln.

Dazu einige Beispiele [1)]:

(1) Verschachtelung von Sequenz und Entscheidung

		1	2
B1	Bedingung a	J	N
A1	Aktion α	X	
A2	Aktion β	X	
A3	Aktion γ		X
A4	AUSGANG	X	X

(2) Verschachtelung von Iterationen



Die Entscheidungstabellentechnik bietet neben der Verschachtelung elementarer Tabellen jedoch noch eine weitere Möglichkeit. komplexe Strukturen aufzubauen. Durch *Kombination* von elementaren Tabellen entstehen größere Tabellen.

---

1) Vgl. Strunz (Verträglichkeit der Strukturierten Programmierung), S. 22

(1) Kombination aus zwei Entscheidungs-Strukturen

		1	2	3	4
B1	Bedingung a	J	J	N	N
B2	Bedingung b	J	N	J	N
A1	Aktion α	X		X	X
A2	Aktion β		X	X	
A3	Aktion γ				X
A4	AUSGANG	X	X	X	X

(2) Kombination aus Entscheidung und Iteration

		1	2	3
B1	Bedingung c	J	J	N
B2	Bedingung d	J	N	-
A1	Aktion δ	X		
A2	Aktion ε		X	
A3	Wiederholung	X	X	
A4	AUSGANG			X

Die durch Kombination elementarer Entscheidungstabellen entstandenen größeren Entscheidungstabellen lassen sich durch Eindeutigkeitsprüfung und Vollständigkeitsprüfung hinreichend verifizieren, so dass diese Alternative des Aufbaus komplexer Programmeinheiten in dieser Beziehung nicht hinter der Verschachtelung von elementaren Strukturen zurücksteht.

Eine durch die Erkenntnisse der Strukturierten Programmierung angereicherte Entscheidungstabellentechnik erscheint - dies sei abschließend festgestellt - leistungsfähiger als die "reine" Strukturierte Programmierung einerseits aber auch leistungsfähiger als die traditionelle Entscheidungstabellentechnik.

5.1.2.3 Gedanken zur Integration von Modularer Programmierung, Strukturierter Programmierung und Entscheidungstabellentechnik

Wenn mehrere Programmiermethoden zur Verfügung stehen, die sowohl Stärken als auch Schwächen haben, liegt der Gedanke nahe, sie so zu kombinieren, dass die Schwächen einer Methode durch die Stärken einer anderen Methode kompensiert werden. Voraussetzung ist jedoch, dass sich die Einzelmethoden miteinander vereinbaren lassen. Letzteres wurde bereits für das Verhältnis Modulare Programmierung zu Entscheidungstabellentechnik und Strukturierte Programmierung zu Entscheidungstabellentechnik nachgewiesen. Es verbleibt eine Untersuchung der Vereinbarkeit von Modularer Programmierung mit Strukturierter Programmierung.

Bereits bei der Darstellung der Strukturierten Programmierung wurde darauf hingewiesen, dass eines ihrer Merkmale, die Top-down-Programmierung, auch in der Modularen Programmierung bekannt ist. Ein weiteres Merkmal, das Konzept der Abstraktionsebenen, wäre auch im Rahmen einer Modularen Programmierung unverändert anwendbar, da es sich dabei um ein allgemeingültiges Konzept der Zerlegung komplexer Systeme handelt. Das wichtigste Merkmal schließlich, die elementaren Ablaufstrukturen, findet in der Modularen Programmierung kein Gegenstück. In der Modularen Programmierung ist es üblich, jeden Modul als "black box" zu betrachten. Hinweise zum problembezogenen internen Aufbau eines Moduls sind nicht Bestandteile der Prinzipien Modularer Programmierung.

Wenn somit auch einer Kombination von Modularer Programmierung und Strukturierter Programmierung kein wesentlicher Einwand entgegensteht, kann versucht werden, die Vorteile einer Integration aller drei erwähnten Methoden zu begründen.

Da alle drei Methoden in ihren Auswirkungen nicht nur auf die Phase der systemtechnischen Implementierung beschränkt bleiben, sondern bereits den systemtechnischen Entwurf beeinflussen, sei kurz auf die Vorteile der Methodenkombination in diesem Bereich eingegangen.

Basierend auf den Informationsverarbeitungsaufgaben, die während des organisatorischen Entwurfs definiert worden sind, werden in der Phase des systemtechnischen Entwurfs die Datenverarbeitungsprozesse global gestaltet. Die Entscheidungstabellentechnik ist in dieser Phase nur partiell anwendbar. Komplexe Systemteile lassen sich mit Hilfe der Entscheidungsmatrix analysieren und Bedingungs-/Aktionskonzentrationen ermitteln.

Die Strukturierte Programmierung empfiehlt, in dieser Phase das Konzept der Abstraktionsebenen anzuwenden und schlägt außerdem vor, den Entwurf in Top-down-Richtung durchzuführen.[1)] Beide Gestaltungsprinzipien sind relativ abstrakt. Sie können zwar bei jedem Datenverarbeitungsproblem angewendet werden, lassen es jedoch an konkreten Anwendungsregeln fehlen, so dass der individuelle Einfluss der Fähigkeiten des Systemplaners die Qualität des Entwurfs noch stark mitbestimmt.

Die Modulare Programmierung scheint in der Entwurfsphase die konkretesten Gestaltungshinweise zu geben. Sie empfiehlt nicht nur den Aufbau eines Systems in Form einer Modulhierarchie, wobei die Beziehungen der Moduln minimiert werden, sondern sie gibt auch konkrete Hinweise, wie bestimmte, immer wieder auftretende Datenverarbeitungsfunktionen modular zu untergliedern sind.

Häufig findet sich beispielsweise der Hinweis, einen Steuermodul zu gestalten, der die Aufrufe der anderen Moduln überwacht.[2)] Ebenso sollen alle Eingabe-/Ausgabeoperationen für eine Datei in einem Modul zusammengefasst werden.[3)]

---

1) Letztere Empfehlung ist, wie bereits dargestellt, von der modularen Programmierung in die Strukturierte Programmierung übernommen worden.

2) Vgl. Lee (Modulare Programmierung). S. 534; Cohen (Defining the module). S. 34; Maynard (Modular programming). S. 15

3) Vgl. Maynard (Modular programming). S. 15 f.; Lee (Modulare Programmierung). S. 534

Auch die Plausibilitätsprüfungen zu einer Eingabedatei oder auch zu mehreren, untereinander logisch verbundenen Dateien eignen sich zur Konzentration in einem Modul. [1)] Weiterhin sollen die Prozesse zur Erarbeitung der Ausgabedaten und zur Datenausgabe auf unterschiedliche Moduln verteilt werden.[2)] Diese Liste der Empfehlungen ließe sich weiter fortsetzen.[3)]

Während in der Phase des systemtechnischen Entwurfs der Schwerpunkt der Aktivitäten mehr bei der Entwicklung der Makrostruktur des Systems liegt und deshalb die Modulare Programmierung intensiver als Strukturierte Programmierung und Entscheidungstabellentechnik genutzt werden kann, kehrt sich dieses Verhältnis bei der Gestaltung der Mikrostruktur - also des internen Aufbaus jedes Moduls - in der Implementierungsphase um.

Die Modulare Programmierung nimmt nur noch insoweit auf die Gestaltung eines Moduls Einfluss, als sie die Verfahren der Modulverknüpfung reglementiert.[4)] Strukturierte Programmierung und Entscheidungstabellentechnik dagegen erlauben auch eine Strukturierung sämtlicher anderer Anweisungen des Moduls. Es wurde bereits ausgeführt, wie Strukturierte Programmierung und Entscheidungstabellentechnik kombiniert werden können. Aus praktischen Erwagungen heraus wird die Entscheidungstabelle immer dann Priorität erhalten, wenn sich die Bedingungen häufen. Anweisungsfolgen, die keine oder nur wenige verstreut angeordnete Bedingungsabfragen enthalten, werden nach den Regeln der Strukturierten Programmierung gestaltet. Wann eine Entscheidungssituation noch mit Hilfe von IF-THEN-ELSE oder CASE beschrieben werden soll und wann die Anwendung der Entscheidungstabelle sinnvoller ist, lässt sich nicht exakt definieren. Hier werden noch praktische Erfahrungen abzuwarten sein.[5)]

Auch ohne die vorgeschlagene Methodenintegration empirisch getestet zu haben, lässt sich vermuten, dass die wesentlichen positiven Auswirkungen der einzelnen Methoden auch innerhalb der Methodenintegration erhalten bleiben. Die Schwächen der Einzelmethoden dagegen - in der Regel handelt es sich dabei um Lücken im Anwendungsprofil - werden innerhalb der integrativen Anwendung verringert oder gar eliminiert.

---

1) Vgl. Lee (Modulare Programmierung), S. 534
2) Vgl. Canning (Modular COBOL programming), S. 11
3) Zu weiteren Modularisierungshinweisen vgl. Canning (Modular COBOL programming), S. 10 ff.
4) Vgl. Maynard (Modular programming), S. 71 ff.
5) Auch die Eignung der angewendeten Programmiersprache zur Bildung der elementaren Ablaufstrukturen wird sich dabei auswirken.

5.1.3 Der Einfluss der Entscheidungstabellen-Anwendung auf die Programm-Übersetzung

Die Teilaufgabe "Übersetzung" im Rahmen der systemtechnischen Implementierung ist eine vollständig automatisierbare Aufgabe. In der Regel verfügt ein Rechner über einen Assemblierer zur Übersetzung von Programmen, die in einer maschinenorientierten Sprache abgefasst sind, und mehrere Kompilierer für Programme in verschiedenen problemorientierten Programmiersprachen.

Die Verwendung von Entscheidungstabellen in einem Programm stellt eine zusätzliche Anforderung an die Übersetzung: Die Quellprogrammanweisungen sind teilweise in Form der Entscheidungstabelle abgefasst. Vor Übergabe des Programms an den Assemblierer oder Kompilierer müssen diese tabellarischen Strukturen in lineare Anweisungsfolgen aufgelöst werden, sofern keine Fähigkeiten zur Behandlung von Entscheidungstabellen im Übersetzer implementiert sind. Da letzteres die Regel ist, wird die Wirksamkeit der Entscheidungstabellenanwendung in der Phase der Systemimplementierung auch durch die Art des Verfahrens beeinflusst, das zur Auflösung der Entscheidungstabellen verwendet wird. In den folgenden Unterabschnitten werden die heute in der Praxis angewendeten Verfahren diskutiert.

5.1.3.1 Konzepte der Behandlung von Entscheidungstabellen in einem Programm

Sofern die Auflösung von Entscheidungstabellen nicht automatisch erfolgt, sondern manuell vorgenommen werden muss, stellt sich die Frage, in welcher Phase der Codierung das erfolgen soll. Die Praxis kennt zwei Vorgehensweisen. Entweder wird der Inhalt der während des systemtechnischen Entwurfs erstellten Entscheidungstabellen zunächst in der ausgewählten Programmiersprache codiert und erst danach erfolgt die Auflösung des tabellarischen Formats, oder Codierung und Auflösung der Tabelle erfolgen in einem Arbeitsgang. In der Anwendung der Entscheidungstabellentechnik erfahrene Programmierer bevorzugen überwiegend die zweite Alternative. Möglich wird die zweite Alternative nur dadurch, dass relativ einfach anwendbare Umsetzungsverfahren zur Verfügung stehen, die dem Programmierer bei nicht zu großen Tabellen erlauben, sich auf die Übertragung von Bedingungen, Aktionen und Anzeigern in die formale Sprache zu konzentrieren. Durch die gleichzeitig durchzuführende Auflösung der tabellarischen Struktur wird er nur unwesentlich belastet.

Gerade weil die Auflösung von Entscheidungstabellen ein relativ einfach auszuführender schematischer Vorgang ist, erscheint es nicht verwunderlich, dass die Geschichte der Entscheidungstabellentechnik eine enge Verflechtung der Entwicklung der theoretischen Grundlagen der Entscheidungstabellentechnik mit der experimentellen Untersuchung der Möglichkeiten ihrer automatischen Verarbeitung erkennen lässt. Dabei wurde zuerst die Technik der Interpretation angewendet. Die ersten Versionen des Interpretierers TABSOL wurden von Mitarbeitern der General Electric für die Anlagen IBM 702, IBM 305, IBM 650 und IBM 704 erstellt.[1)] Bei Anwendung eines Interpretierers bleibt das tabellarische Format der Entscheidungstabelle unverändert. Zur Ausführungszeit interpretieren die Unterprogramme des Interpretierers die einzelnen logischen Elemente der Entscheidungstabelle und führen sie unmittelbar aus.

Interpretierer haben Vor- und Nachteile.[2)] Die Entscheidungstabellen können als Datei betrachtet und relativ einfach aktualisiert werden. Da die Tabellen in ihrem kompakten Format erhalten bleiben, benötigen sie nur wenig Hauptspeicherplatz. Interpretierer benötigen jedoch eine hohe Ausführungszeit. Jede Entscheidungstabelle wird dem gleichen Aus-

---

1) Vgl. Kavanagh (TABSOL), S. 124
2) Vgl. zum folgenden Wright (Decision table processors), S. 42

führungsverfahren unterworfen, besondere Optimierungen in Abhängigkeit von syntaktischen Merkmalen der Entscheidungstabelle sind nicht möglich. Im Vergleich zu anderen Verfahren der automatischen Ausführung von Entscheidungstabellen gilt die Interpretation als so aufwendig, dass reine Entscheidungstabelleninterpretierer keine praktische Bedeutung erlangt haben.

Übersetzer für Entscheidungstabellen dagegen sind heute in der Praxis sehr viel weiter verbreitet. Jeder Übersetzer übersetzt Anweisungen einer Quellsprache in Anweisungen einer Zielsprache. Beispielsweise übersetzen Kompilierer Anweisungen problemorientierter Programmiersprachen in Maschinensprache. Quellsprache und Zielsprache sind formale Sprachen mit klar definierten Syntaxregeln, so dass die Übersetzung vollautomatisch erfolgen kann. Bei der Übersetzung von Programmen, die Entscheidungstabellen enthalten, gilt es jedoch, zwei Quellsprachen zu übersetzen. Die Entscheidungstabellenstruktur ist in eine lineare Struktur zu verwandeln, und die sich aus Bedingungen, Aktionen und Entscheidungsregeln ergebenden Anweisungsfolgen sind in Maschinensprache zu übersetzen. Ein Entscheidungstabellenübersetzer kann somit seine Zielanweisungen in Maschinensprache generieren, oder er kann sich auf die Übersetzung der Entscheidungstabellenstruktur beschränken und die innerhalb der Entscheidungstabellen verwendete Sprache als Zielsprache der Übersetzung benutzen. Im letzteren Fall fungiert der Entscheidungstabellenübersetzer als *Vorübersetzer*, da seine Zielsprache im gesamten Übersetzungsprozess den Charakter einer Zwischensprache hat.

Das Prinzip der *Vorübersetzung* hat sich in der Praxis durchgesetzt. Eine vom Verfasser 1970 durchgeführte Untersuchung ergab, dass von 24 ermittelten Entscheidungstabellenübersetzern nur 2 Maschinensprache als Zielsprache erzeugten. Alle anderen verwendeten COBOL, FORTRAN oder PL/I als Zielsprache.[1)] Dem Verfasser ist in den letzten sieben Jahren kein neuentwickelter Entscheidungstabellenübersetzer mit Maschinensprache als Zielsprache bekannt geworden.

Die eindeutige Bevorzugung des Vorübersetzungsprinzips bei der Implementierung von Entscheidungstabellenübersetzern kann auf einen einzigen Grund zurückgeführt werden. Fast alle der heute eingesetzten

---

1) Vgl. Strunz (Voruntersuchungen), S. 32 ff.

Entscheidungstabellenübersetzer sind auf der Basis der am weitesten verbreitenden problemorientierten Programmiersprachen COBOL, FORTRAN und PL/I entwickelt worden.[1)] COBOL und FORTRAN sind standardisierte Sprachen, PL/I wird derzeit standardisiert. Aus diesem Grund sind fast alle Rechner-Hersteller gezwungen, Kompilierer für diese Sprachen bereitzustellen. Da die Entwicklung eines Kompilierers etliche Mannjahre Aufwand verursacht,[2)] ist es für jeden Implementator eines Entscheidungstabellenvorübersetzers schon ein Gebot wirtschaftlicher Vernunft, sich auf die Entwicklung eines *Vorübersetzers* zu beschränken und für den zweiten Übersetzungsschritt vorhandene Kompilierer zu benutzen.

Die Entscheidung zum Einsatz eines Entscheidungstabellenvorübersetzers bringt dem Entscheidungstabellenanwender den Nachteil einer Verlängerung des Übersetzungsvorgangs.[3)] Verglichen mit der Anwendung konventioneller Programmerstellungstechniken ist das jedoch nicht als Quelle von Unwirtschaftlichkeiten zu betrachten. Die Anzahl von Übersetzungsläufen während der Programmerstellung geht aufgrund der bei intensiver Anwendung der Entscheidungstabellentechnik erheblich besseren Möglichkeiten einer strukturellen Programmverifikation stark zurück.

Das Prinzip der *direkten* Übersetzung von Entscheidungstabellen wird wieder Übergewicht bekommen, wenn die Entscheidungstabelle als Sprachelement in die bekannten Programmiersprachen aufgenommen wird. Zumindest bei COBOL ist das zu erwarten. Eine Mehrheit der Mitglieder des Programming Language Committee der CODASYL sprach sich bereits dafür aus.[4)] Aufgrund des langwierigen Sprachentwicklungs- und Standardisierungsprozesses kann mit der Verfügbarkeit von COBOL -Kompilierern mit dem zusätzlichen funktionalen Modul "decision table processing" wohl erst Anfang der achtziger Jahre gerechnet werden.

5.1.3.2 Algorithmentypen zur Übersetzung von Entscheidungstabellen

Ob die Übersetzung von Entscheidungstabellen manuell oder maschinell erfolgt, in beiden Fällen wird ein Verfahren benötigt, das die tabel-

---

1) Für Assembler-Sprachen werden Macros zur Entscheidungstabellendefinition und -übersetzung angeboten. Vgl. beispielsweise Harris (Decision table macros).
2) Beispielsweise wurde bei einer Befragung von Herstellern der Aufwand zur Entwicklung eines COBOL -Kompilierers mit 20 bis 30 Mannjahren angegeben. Vgl. Strunz (COBOL), S. 322
3) Vgl. Wright (Decision table processors), S. 43
4) Vgl. Rabin (DTTG Progress Report), S. 1

larische Struktur in eine lineare Struktur umzuwandeln gestattet, ohne dabei den logischen Inhalt der Entscheidungstabelle zu verändern. Im Folgenden ersten Unterabschnitt sollen manuelle Codierverfahren erläutert und kritisch untersucht werden, im zweiten Unterabschnitt sollen Übersetzeralgorithmen diskutiert werden.

5.1.3.2.1 Verfahren zur manuellen Codierung von Entscheidungstabellen

Zur manuellen Codierung von Entscheidungstabellen eignen sich drei Verfahren: die Codierung "Regel-führ-Regel", das Entscheidungsbaumverfahren und die Veinott-Methode. In den folgenden Unterabschnitten werden die zugrundeliegenden Algorithmen erläutert. Ein Vorschlag zur speicherplatzminimierenden Codierung der Aktionsfolgen einer Entscheidungstabelle soll die Erörterung manueller Codierverfahren beschließen.

5.1.3.2.1.1 Codierung "Regel-für-Regel"

Die Codierung "Regel-für-Regel"[1)] gilt als einfachstes Verfahren der Umsetzung einer Entscheidungstabelle in ein Programm. Dabei wird jede Regel als selbständige Einheit betrachtet und in eine Anweisungsfolge übersetzt, in deren erstem Teil die Bedingungsabfragen und deren zweiten Teil die Aktionen codiert sind.

Aus Gründen der Laufzeitoptimierung ist zu empfehlen, vor Codierung der Entscheidungsregeln die Häufigkeiten ihrer voraussichtlichen Ausführung zu bestimmen. Die Ausführungszeit wird minimiert, wenn die Regeln nach abnehmender Häufigkeit ihrer Ausführung codiert werden.

---

1) Vgl. Fergus (Decision tables), S. 102

Beispiel:

		1	2	3	4	5	ELSE
B1	Bedingung a	N	N	-	-	J	
B2	Bedingung b	N	J	N	J	N	
B3	Bedingung c	J	-	N	N	J	
B4	Bedingung d	N	J	-	N	J	
A1	Aktion α	α1	α2	α3	α4	α5	α6

Struktur der Codierung:

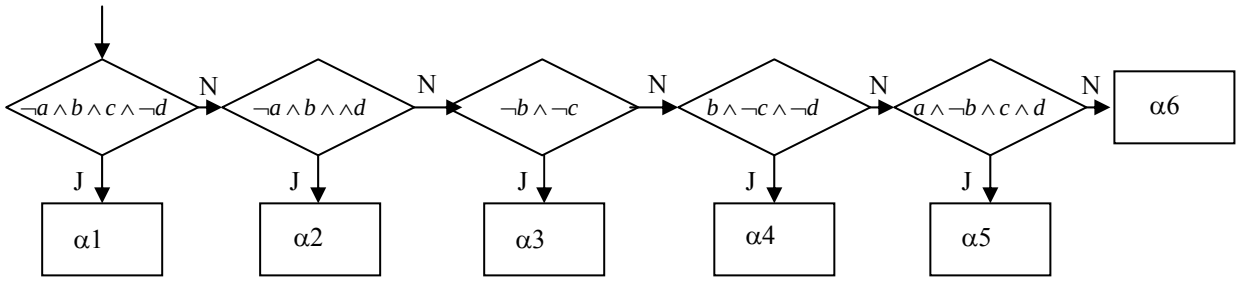


Abb. 15: Struktur der Codierung beim Regel – für –Regel -Verfahren

Bessere Laufzeitergebnisse sind zu erwarten, wenn nach jeder Bedingungsprüfung entschieden wird, ob die Regel noch zutreffen kann, und im negativen Fall sofort zum Anfang der nächsten Regel gesprungen wird,[1)] wie das folgende Schaubild zeigt.

		1	2	3	4	5	ELSE
B1	Bedingung a	N	N	-	-	J	
B2	Bedingung b	N	J	N	J	N	
B3	Bedingung c	J	-	N	N	J	
B4	Bedingung d	N	J	-	N	J	
A1	Aktion α	α1	α2	α3	α4	α5	α6

1) Vgl. dazu Press (Conversion of decision tables), S. 385

Struktur der Codierung:

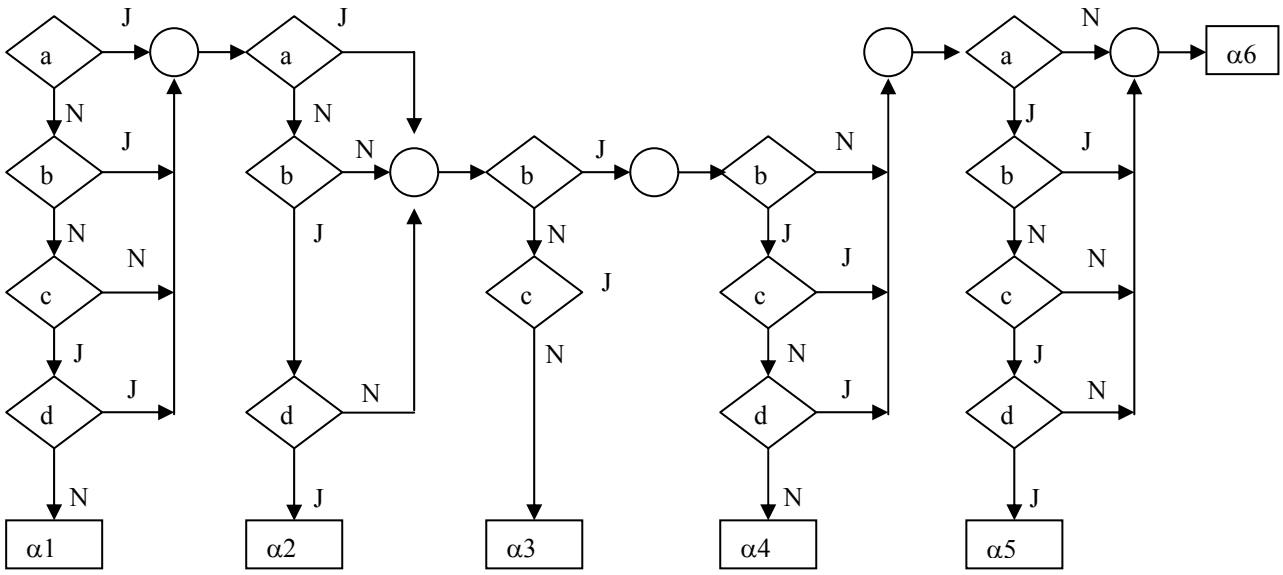


Abb. 16: Struktur der Codierung beim optimierenden Regel –für-Regel-Verfahren

Charakteristisch für das "Regel-für-Regel"-Verfahren ist, dass jeder Bedingungsanzeiger mit Ausnahme des Indifferenzanzeigers zur Codierung einer Bedingungsprüfung führt.[1)] Im Übrigen werden Indifferenzanzeiger und Ausschlußanzeiger gleich behandelt. Beide Fälle führen zur Auslassung der Bedingungsprüfung.

Die Codierung erweiterter oder gemischter Entscheidungstabellen bereitet ebenfalls keine Schwierigkeiten. Der in der untersuchten Regel verlangte Bedingungszustand wird abgeprüft. Falls er zutrifft, wird innerhalb der Regel weiter geprüft, andernfalls wird zum Beginn der nächsten Regel verzweigt.

Auch Mehrtreffer-Entscheidungstabellen lassen sich nach dem "Regel -für -Regel" -Verfahren codieren. Allerdings ist eine kleine Modifikation erforderlich. Nachdem eine Regel als zutreffend erkannt wurde, wird

1) Vgl. Press (Conversion of decision tables), S. 385

diese Information in irgendeiner Form gespeichert und mit der Prüfung der nächsten Regel fortgefahren. Zusätzlich muss eine geeignete Codierung der Aktionen gefunden werden, die vor der Aktionsausführung die Vereinigung der Aktionsfolgen mehrerer Regeln erlaubt.

5.1.3.2.1.2 Entscheidungsbaum-Verfahren

Das Entscheidungsbaumverfahren setzt die bedingungsorientierte Zergliederung der Entscheidungstabelle voraus.[1)] Die Codierung spiegelt dann nichts anderes als einen Entscheidungstabellenverbund wieder, dessen Einzeltabellen jeweils eine Bedingung enthalten und der in der Art eines Entscheidungsbaumes strukturiert ist. Da die Anzahl der Knoten dieser Entscheidungsbaumstruktur Hauptspeicherbedarf und Laufzeit des entstehenden Programms bestimmt, sind verschiedene Verfahren vorgeschlagen worden, die die Baumstruktur minimieren sollen.

Egler empfiehlt, die Entscheidungstabelle vor der Codierung zu sortieren: Bedingungszeilen werden nach zunehmender Anzahl von Indifferenzanzeigern und die Regeln nach Maßgabe der Bedingungsanzeiger lexikographisch geordnet.[2)]

Montalbano wies in seiner Kritik an Eglers Algorithmen nach, dass nicht in allen Fällen ein Minimum an Bedingungsprüfungen erzielt werden kann.[3)] Dennoch wird die von Egler vorgeschlagene Optimierung in der Praxis angewendet, da alle anderen bisher vorgeschlagenen "besseren" Optimierungsverfahren zu kompliziert für die manuelle Codierung sind. Die in der ELSE-Regel abgedeckte Logik - von Egler nicht berücksichtigt - führt dazu, dass verschiedene Zweige des entstehenden Entscheidungsbaumes zu den für die ELSE-Regel vorgesehenen Aktionen führen müssen. Die ELSE-Zweige ergeben sich zwangsläufig während des Zergliederungsprozesses, so dass die mit der ELSE-Regel abgedeckten Bedingungskonstellationen nicht vor der Codierung explizit dargestellt zu werden brauchen.[4)]

---

1) Die Grundidee zu diesem Verfahren geht offenbar auf Montalbano zurück. Vgl. Montalbano (Decision tables), S. 129 ff.
2) Vgl. Egler (Converting logic table conditions), S. 510 ff.
3) Vgl. Montalbano (Egler's procedure), S. 1
4) Vgl. dazu Pollack (Conversion of limited-entry decision tables), S. 678

Beispiel:

		1	2	3	4	5	ELSE
B1	Bedingung a	N	N	-	-	J	
B2	Bedingung b	N	J	N	J	N	
B3	Bedingung c	J	-	N	N	J	
B4	Bedingung d	N	J	-	N	J	
A1	Aktion α	α1	α2	α3	α4	α5	α6

Nach Sortierung:

		1	2	3	4	5	ELSE
B1	Bedingung b	J	J	N	N	N	
B2	Bedingung c	N	-	J	J	N	
B3	Bedingung d	N	J	J	N	-	
B4	Bedingung a	-	N	J	N	-	
A1	Aktion α	α1	α2	α3	α4	α5	α6

Der Programmablaufplan der Codierung wird durch das folgende Schaubild verdeutlicht.

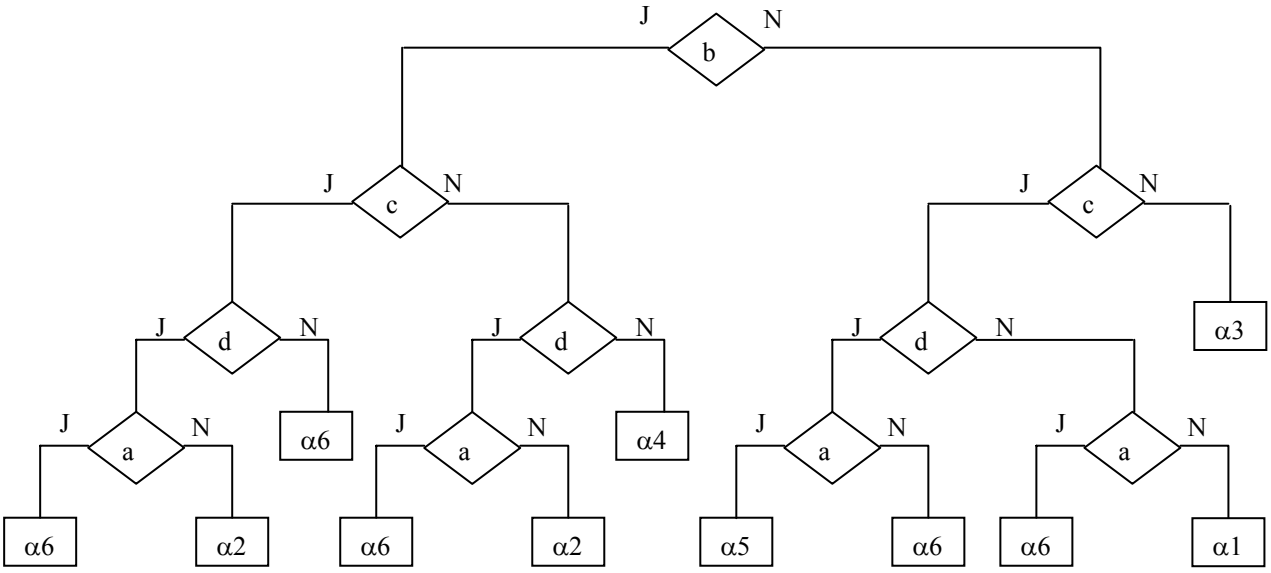


Abb. 17: Struktur der Codierung beim Entscheidungsbaumverfahren von Egler

Wird einer Empfehlung von Hughes, Shank, Stein gefolgt, dann wird die sich nach der Umsortierung ergebende Bedingungsfolge nicht schematisch beibehalten, sondern innerhalb der Zweige kann die Reihenfolge der Bedingungen nochmals verändert werden, um Bedingungszeilen mit Indifferenzanzeiger so weit wie möglich nach unten zu verlagern.[1)] Im gewählten Beispiel würde sich dann die Abfragestruktur im Ja-Zweig der Bedingung b wie folgt verändern:

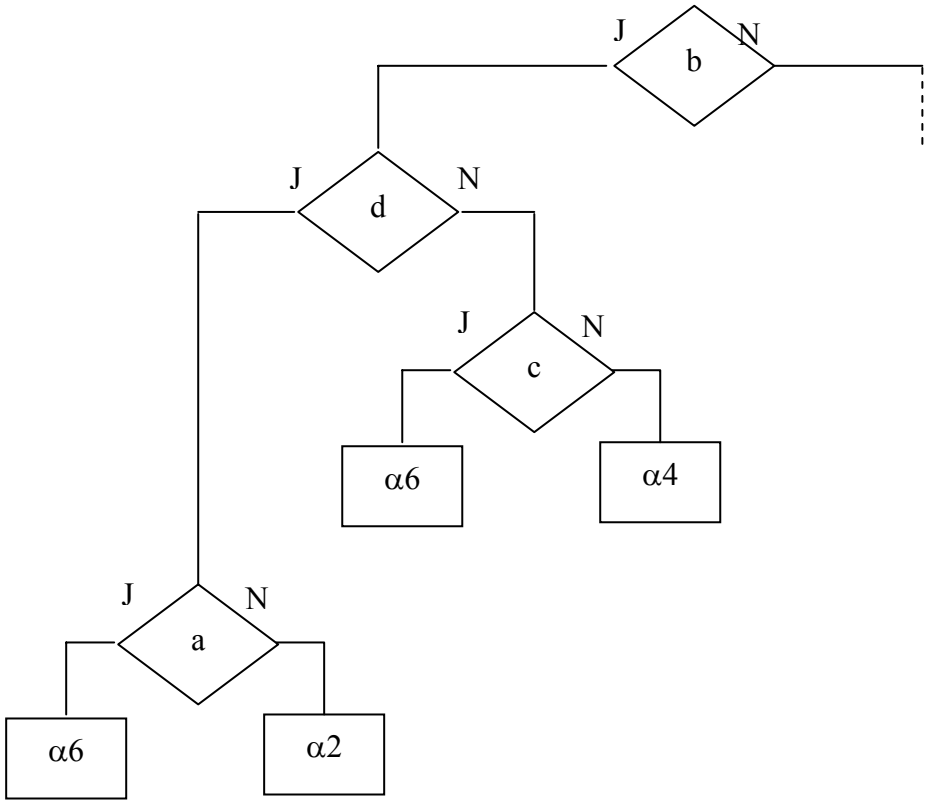


Abb. 18: Optimierte Entscheidungsbaumstruktur nach Hughes, Shank, Stein

Das Resultat lässt sowohl weniger Hauptspeicherbedarf als auch geringere Laufzeit erwarten.

Es ist leicht ersichtlich, dass die Wirksamkeit eines nach dem Entscheidungsbaumverfahren entstandenen Programms stark von der Möglichkeit abhängig ist, die Bedingungsfolge abändern zu können. Bei logisch abhängigen Bedingungen kann eine Änderung der Bedingungsreihenfolge aber aus semantischen Gründen unzulässig sein, der Programmierer muss in diesen Fällen auf eine Optimierung verzichten.

---

1) Vgl. Hughes; Shank; Stein (Decision tables), S. 101 ff.

Das Entscheidungsbaumverfahren ist bei allen Eintreffer-Entscheidungstabellen anwendbar. In der Literatur findet sich häufig der Hinweis, dass der Algorithmus auf begrenzte Entscheidungstabellen angewendet werden sollte, da sich erweiterte oder gemischte Entscheidungsstabellen in begrenzte Tabellen überführen lassen.[1)] Eine direkte Codierung ist jedoch auch möglich, wenn die benutzte Programmiersprache über ein der CASE -Anweisung entsprechendes Sprachelement verfügt. Mehrtreffer–Entscheidungstabellen lassen sich nach dem Entscheidungsbaumverfahren jedoch nicht codieren.

5.1.3.2.1.3 Veinott - Methode

Veinott hat ein weiteres Codierverfahren vorgeschlagen, das sich von den beiden bereits erläuterten Verfahren grundsätzlich unterscheidet.[2)] Er stellt die maximale Anzahl von Bedingungskonstellationen der durch eine Entscheidungstabelle beschriebenen Entscheidungssituation in den Mittelpunkt seiner Überlegungen. Jede dieser Bedingungskonstellationen kann, muss aber nicht zu einer anderen Aktionsfolge führen. Wichtigster Bestandteil seines Algorithmus ist deshalb eine Sprungverteilung, die die Ansteuerung der richtigen Aktionsfolge in Abhängigkeit von der aktuellen Bedingungskonstellation bewirkt. Die einzelnen Teilabschnitte von Veinotts Algorithmus sind:

- Prüfen aller Bedingungen, um die aktuelle Bedingungskonstellation festzustellen;
- Errechnen, welche Regelnummer diese Bedingungskonstellation hatte, wenn die Tabelle in ihre kanonische Normalform verwandelt würde;
- Benutzen dieser Regelnummer zur Ansteuerung der zugeordneten Aktionsfolge.

Die bisher benutzte Beispieltabelle kann nach dem Veinott-Verfahren[3)] in COBOL wie folgt codiert werden:

---

1) Vgl. Pollack (Conversion of limited-entry decision tables), S. 677; Pooch (Translation of decision tables), S. 139;
Press (Conversion of decision tables), S. 388 f.

2) Vgl. Veinott (Programming decision tables), S. 31 ff.

3) Die Errechnung der Regelnummer wurde in einer von Veinotts Vorschlägen abweichender Form codiert, da seine Lösung zu laufzeittaufwendig ist.

Beispiel:

		1	2	3	4	5	ELSE
B1	Bedingung a	N	N	-	-	J	
B2	Bedingung b	N	J	N	J	N	
B3	Bedingung c	J	-	N	N	J	
B4	Bedingung d	N	J	-	N	J	
A1	Aktion α	α1	α2	α3	α4	α5	α6

```
MOVE 16 TO SPRUNG

IF Bedingung a    SUBTRACT  8 FROM SPRUNG.
IF Bedingung b    SUBTRACT  4 FROM SPRUNG.
IF Bedingung c    SUBTRACT  2 FROM SPRUNG.
IF Bedingung d    SUBTRACT  1 FROM SPRUNG.

GO TO ELS ELS ELS ALFA-4 ALFA-5 ELS ALFA-3 ALFA-3
      ALFA-2 ELS ALFA-2 ALFA-4 ELS ALFA-1 ALFA-3 ALFA-3
      DEPENDING ON SPRUNG.


ALFA-l.
ALFA-2
ALFA-3.
```

Die Veinott-Methode liegt an der Grenze dessen, was noch bei manueller Codierung zumutbar ist. Insbesondere ist die Ermittlung der Sprungadressen für den berechneten SPRUNG (GO TO ... DEPENDING ON) kaum möglich, ohne die Entscheidungstabelle tatsächlich in ihre kanonische Normalform zu überführen. Im Übrigen lassen sich auch erweiterte oder gemischte Entscheidungstabellen nach dem Veinott-Verfahren codieren.[1)] Der Aufwand zur Vorbereitung der Codierung größerer Tabellen ist dann jedoch kaum noch zumutbar. Entscheidungstabellen mit Ausschlußanzeiger lassen sich mit der Veinott-Methode nicht codieren.[2)] Auch Mehrtreffer-Entscheidungstabellen sind nach diesem Verfahren nicht codierbar.

---

1) Vgl. Veinott (Programming decision tables), S. 33 f.

2) Die Veinott-Methode setzt voraus, dass alle Bedingungen ohne Einschränkung gemeinsam überprüfbar sind. Der Ausschlußanzeiger zeigt jedoch an, dass eine Bedingung innerhalb einer bestimmten Bedingungskonstellation nicht definiert ist. Um die Veinott-Methode auch bei Ausschlußanzeiger anwenden zu können, müsste sie tabellenbezogen modifiziert werden, was bei manueller Codierung grundsätzlich möglich ist.

5.1.3.2.1.4 Optimierung der Aktionen -Codierung

Wenn mit Hilfe eines der erläuterten Codierverfahren eine Entscheidungsregel identifiziert worden ist, dann können die dieser Entscheidungsregel zugeordneten Aktionen ausgeführt werden. Jede in einer Entscheidungstabelle spezifizierte Aktionsfolge vollständig zu codieren, wäre zwar eine laufzeitminimale, jedoch auch sehr hauptspeicherintensive Lösung. Der Hauptspeicherbedarf kann reduziert werden, ohne dass die Laufzeit merklich zunimmt, wenn Gemeinsamkeiten der Aktionsfolgen analysiert und geschickt zu einer Reduzierung der codierten Aktionen ausgenutzt werden. Dent[1)] hat dazu vorgeschlagen; die Aktionen einer Regel nur so weit zu codieren, bis die verbleibenden Aktionen identisch sind mit der Rest-Aktionsfolge einer anderen Regel. Dent wählt folgendes Beispiel, um diese Art der Optimierung zu veranschaulichen:

	1	2	3	4	5	6
Aktion α	X	X		X	X	
Aktion β	X		X	X		X
Aktion γ	X	X	X		X	
Aktion δ	X	X	X	X	X	X

Das Schaubild zeigt die Struktur der optimierten Aktionencodierung.

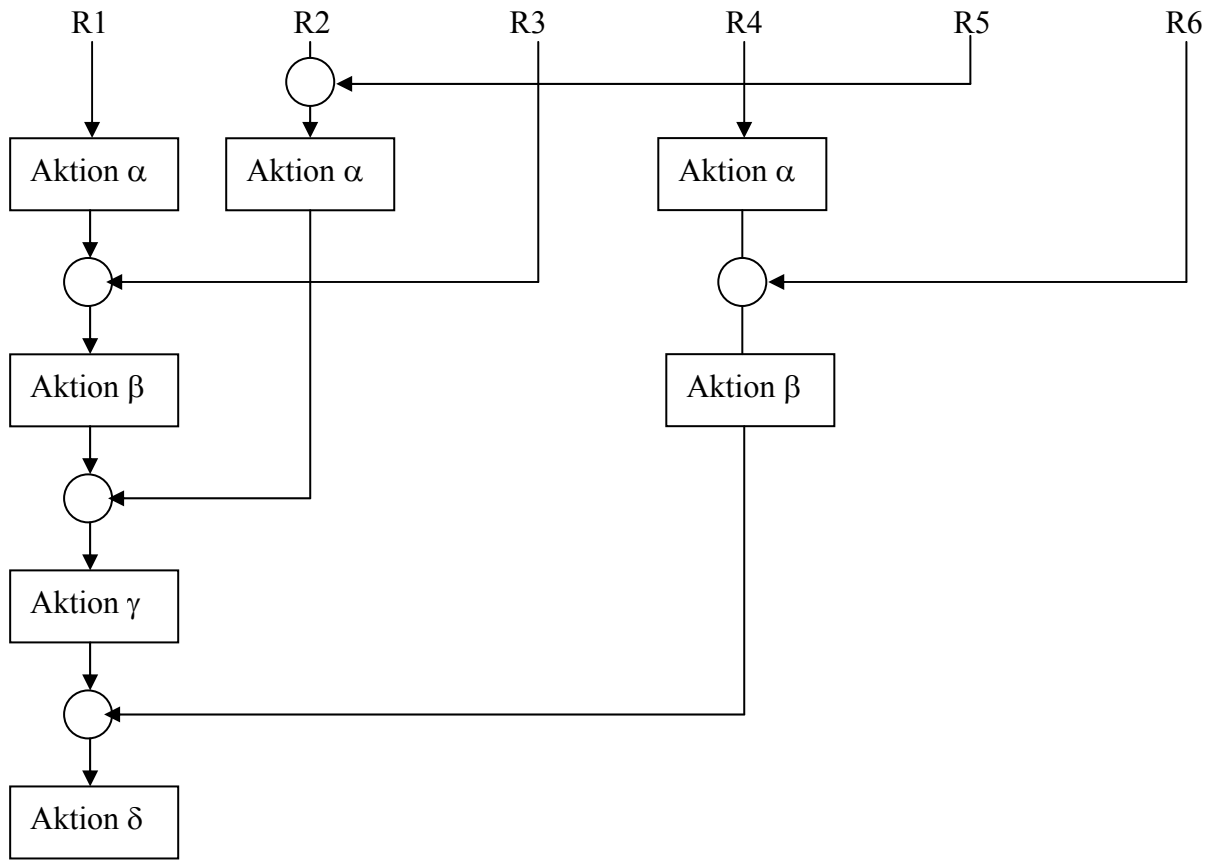


Abb. 19: Struktur der optimierten Aktionencodierung

1) Vgl. Dent (Decision tables), S. 28 f.

Ohne Optimierung wären 18 Aktionen zu codieren gewesen, mit Optimierung sind es nur noch 7. Die optimierte Lösung erfordert jedoch 5 GO-TO -Anweisungen, was sich geringfügig auf die Laufzeit auswirkt.

Wenn die Aktionen einer Entscheidungstabelle sehr hauptspeicheraufwendig sind, kann möglicherweise auch mit PERFORM oder einer entsprechenden Anweisung gearbeitet werden, um Aktionsteilfolgen einer anderen Regel aus der Mitte heraus zu übernehmen.

Jede Optimierung der Aktionencodierung wirkt sich negativ auf die Programmtransparenz aus. Bei Entscheidungstabellen, die häufigen Änderungen unterliegen, sollte möglicherweise auf eine Optimierung verzichtet werden, um der Gefahr von Missverständnissen bei der Änderung der Aktionsfolgen zu begegnen.

5.1.3.2.2 Algorithmen für Entscheidungstabellenübersetzer

Die derzeit bekannten Algorithmen für Entscheidungstabellenübersetzer lassen sich in zwei Kategorien einteilen:

- Algorithmen, die auf dem Entscheidungsbaumverfahren basieren, und
- Algorithmen, die unter der Bezeichnung "Maskenverfahren" zusammengefasst werden können.

Entscheidungstabellenübersetzer-Algorithmen müssen mehr leisten als manuelle Codierverfahren. Während bei der manuellen Codierung von Entscheidungstabellen vorausgesetzt werden kann, dass die Tabellenverifikation bereits erfolgt ist, bleibt das bei der maschinellen Übersetzung Aufgabe des Übersetzers. Für Entscheidungstabellenübersetzer vorgesehene Algorithmen müssen außerdem eine größere Anwendungsbreite besitzen. Sie müssen bei allen Tabellenarten und -größen zufriedenstellende Übersetzungsergebnisse erbringen.

5.1.3.2.2.1 Entscheidungsbaum -Algorithmen

Montalbano hat bereits 1962 die Beschreibung eines übersetzergeeigneten Entscheidungsbaumverfahrens vorgelegt.[1)] Seine Arbeit ist aus mehreren Gründen bemerkenswert. Montalbano erkannte, dass der Übersetzung von Entscheidungstabellen unterschiedliche Zielsetzungen zugrunde liegen können, die sich nicht miteinander vereinbaren lassen. Er beschreibt zwei Algorithmen, die "quick rule method", die eine Minimierung der generierten Verzweigungsanweisungen bewirkt, und die

---

1) Vgl. Montalbano (Tables, flow charts), S. 51 ff.

"delayed-rule method", die die durchschnittliche Anzahl der zur Bestimmung einer Regel ausgeführten Anweisungen minimiert.[1)] Seine beiden Algorithmen sind bereits auf die Verarbeitung erweiterter Entscheidungstabellen ausgerichtet. Das verdient deshalb besondere Beachtung, weil sich nach Montalbano die Algorithmenentwicklung lange Zeit nur noch mit der begrenzten Entscheidungstabelle beschäftigte. Erst 1971 wurde wieder ein Algorithmus zur direkten Verarbeitung von erweiterten Entscheidungstabellen veröffentlicht.[2)] In Unkenntnis der Arbeit Montalbanos bezeichneten die Autoren dieser Arbeit ihren Algorithmus als den einzigen existierenden Algorithmus für die direkte Übersetzung erweiterter Entscheidungstabellen.[3)]

Montalbano hat die Arbeit Pollacks stark befruchtet,[4)] der nach Montalbano die Weiterentwicklung des Entscheidungsbaumverfahrens betrieb und dessen Veröffentlichungen erheblich stärkere Beachtung fanden als Montalbanos Arbeit. Pollack hat sich nur mit begrenzten Entscheidungsstabellen beschäftigt, die direkte Übersetzung von erweiterten Entscheidungstabellen hielt er für überflüssig, da erweiterte Entscheidungstabellen in begrenzte Entscheidungstabellen transformierbar sind.[5)]

In Anlehnung an Montalbano entwickelte auch Pollack zwei Algorithmen, seinen Algorithmus 1, dessen Zielsetzung die Minimierung des Hauptspeicherbedarfs des generierten Programms war,[6)] und seinen Algorithmus 2, der die zur Laufzeit ausgeführten Verzweigungsanweisungen minimierte.[7)] Sprague[8)] kritisierte, dass beide Algorithmen keine echte Minimierung bewirken, und machte einen Verbesserungsvorschlag, der allerdings den Aufwand der Entscheidungstabellenübersetzung nach Meinung Pollacks stark erhöhen wurde.[9)] Eine weitere Verbesserung unter Benutzung von Erkenntnissen der Informationstheorie wurde von Shwayder vorgeschlagen.[10)] Verhelst gelang es, das Entscheidungsbaumverfahren nochmals zu verbessern.[11)]

---

1) Vgl. Montalbano (Tables, flow charts), S. 54 ff.
2) Vgl. Woods; Hawes (Code generation), S. 74 ff.
3) Vgl. Woods; Hawes (Code generation), S. 75
4) Pollack selbstgibt an, Montalbanos Algorithmen als Basis für seine Entwicklungen benutzt zu haben. Vgl. Pollack (Conversion of limited-entry decision tables), S. 679
5) Vgl. Pollack (Conversion of limited-entry decision tables), S. 677. Diese Ansicht bekräftigte Pollack noch in einem 1972 mit dem Verfasser geführten Gespräch.
6) Vgl. Pollack (Conversion of limited-entry decision tables), S. 679 f.
7) Vgl. Pollack (Conversion of limited-entry decision tables), S. 681 f.
8) Vgl. Sprague (Storage space), S. 319
9) Vgl. Pollack (Stellungnahme zu Spragues Einwand), S. 320
10) Vgl. Shwayder (Modification to Pollack's algorithm), S. 69 ff.
11) Vgl. Verhelst (Conversion of limited-entry decision tables), S. 974 ff.

Allgemein wird heute anerkannt, dass die Entwicklung optimierender entscheidungsbaumorientierter Entscheidungstabellen-Algorithmen mit der Arbeit von Reinwald und Soland ihren Höhepunkt fand.[1)] Reinwald und Soland lösen sich von der noch bei Pollack zu findenden Vorstellung, dass ein Algorithmus nur auf der Grundlage der Entscheidungstabellensyntax operieren kann. Sie benutzen Informationen über das Verhalten der Entscheidungstabelle zur Ausführungszeit, um eine optimale Codegenerierung zu erzielen.

Die folgenden Ausführungen sollen sich auf den ersten Algorithmus von Reinwald und Soland beschränken, [2)] der die minimale Laufzeit des entstehenden Programms garantiert. Zur Erzielung minimalen Hauptspeicherbedarfs scheinen die im nächsten Unterabschnitt erläuterten Maskenverfahren geeigneter zu sein.

Reinwald und Soland, die sich auf die Behandlung begrenzter Entscheidungstabellen beschränken, bestimmen die Laufzeit eines aus einer Entscheidungstabelle entstehenden Programms mit Hilfe folgender Einflussfaktoren:

W_j^r ist die Wahrscheinlichkeit, in der die Regel j bei der Ausführung der Entscheidungstabelle zutrifft;

A_i ist der Aufwand (Maschinenzeit), der zur Prüfung der Bedingung i entsteht;

W_i^b ist die Wahrscheinlichkeit, dass die Bedingung i erfüllt ist, wenn sie innerhalb einer Regel als indifferent ausgewiesen ist.

Zur Erläuterung des Algorithmus soll eine Beispieltabelle benutzt werden, die mit allen erforderlichen Zusatzinformationen versehen ist:[3)]

---

1) Vgl. Reinwald; Soland (Conversion of limited-entry decision tables I), S. 339 ff.; Reinwald; Soland (Conversion of limited-entry decision tables II), S. 742 ff.

2) Zugrunde gelegt wird eine spätere Arbeit Reinwalds, in der einige der ursprünglichen Restriktionen des Algorithmus aufgehoben sind. Vgl. Reinwald (Parsing decision tables into computer programs).

3) Vgl. dazu Reinwald (Parsing decision tables into computer programs), S. 11

		A_i	W_i^b	R1	R2	R3	R4	R5
				W_j^r 0,3	0,3	0,2	0,1	0,1
B1	Bedingung a	10	0,5	J	N	-	J	N
B2	Bedingung b	10	0,5	J	-	N	N	J
B3	Bedingung c	10	0,5	-	J	N	J	N

Reinwald und Soland begründen nun, dass sich die Zeit, die benötigt wird, um zu einer aktuellen Bedingungskonstellation die zutreffende Regel herauszufinden, aus zwei Bestandteilen zusammensetzt: Der Zeit zum Beweis, dass die Bedingungen den von der Regel geforderten Zustand aufweisen, und der Zeit zur Prüfung von Bedingungen, die sich in Bezug auf die ausgewählte Regel indifferent verhalten.[1)]

Im ersten Fall soll von "Beweisaufwand" gesprochen werden, im zweiten Fall von "Suchaufwand". Wenn beispielsweise bei der oben abgebildeten Entscheidungstabelle mit der Prüfung der Bedingung a begonnen wird und es stellt sich später heraus, dass die Regel 2 zutrifft, dann war diese Bedingungsprüfung notwendig und der entstandene Aufwand fällt in die Kategorie "Beweisaufwand". Stellt sich jedoch später heraus, dass die Regel 3 zutrifft, dann muss der zur Prüfung der Bedingung a entstandene Aufwand als "Suchaufwand" bezeichnet werden.

Zielsetzung eines laufzeitoptimierenden Algorithmus muss es nun sein, die Reihenfolge der Bedingungsprüfungen so zu bestimmen, dass der Suchaufwand insgesamt minimiert wird. Der Beweisaufwand dagegen lässt sich nicht vermindern.[2)]

Für jede Bedingung der Entscheidungstabelle kann nun auf der Basis ihres Prüfaufwandes und ihrer Regelwahrscheinlichkeiten der zu erwartende Beweisaufwand und der zu erwartende Suchaufwand bestimmt werden.[3)]

Der *Beweisaufwand* ergibt sich aus dem Prüfaufwand, der mit der Summe der Regelwahrscheinlichkeiten gewichtet wird. Die Summe der Regelwahrscheinlichkeiten wird aus den Wahrscheinlichkeiten derjenigen Regeln gebildet,

---

1) Vgl. Reinwald (Parsing decision tables into computer programs), S. 10
2) Val. Reinwald (Parsing decision tables into computer programs), S. 10
3) Vgl. zum folgenden Reinwald (Parsing decision tables into computer programs), S. 10 ff.

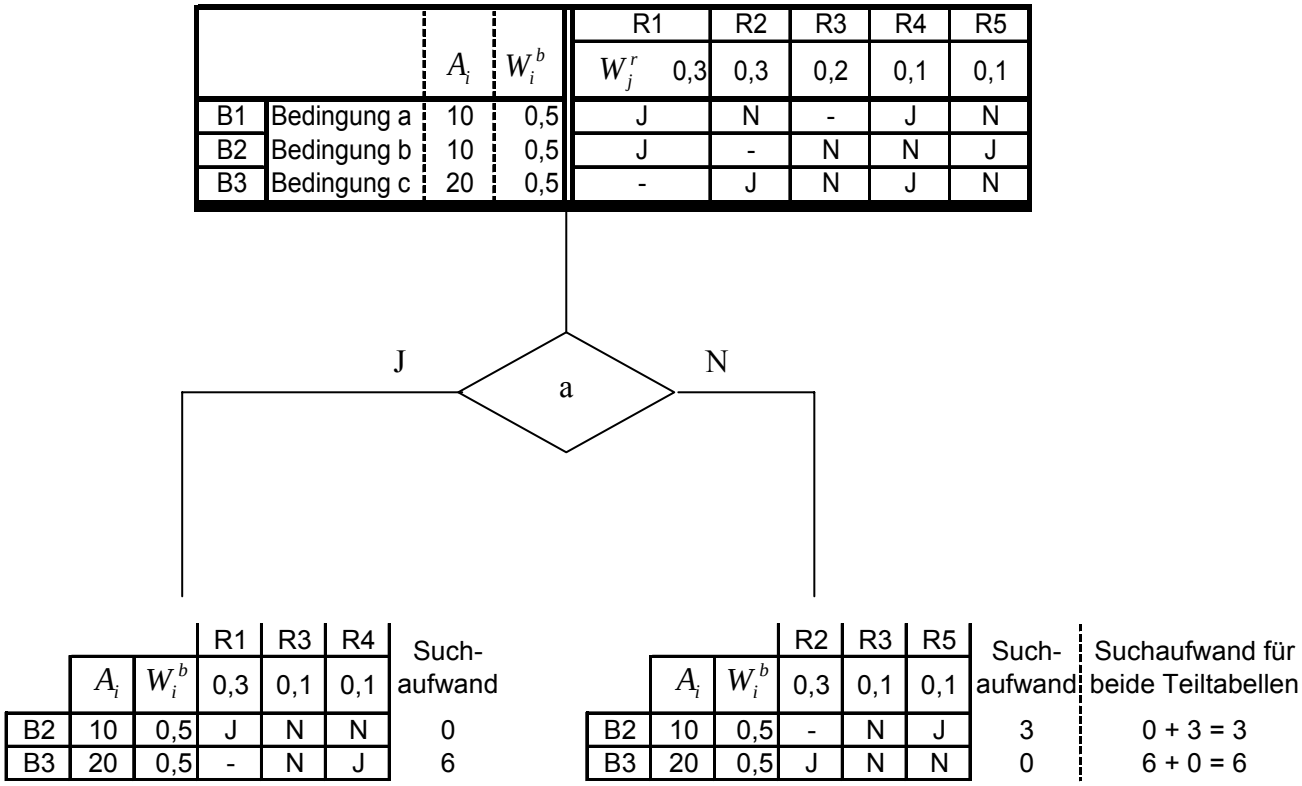
die verlangen, dass die betreffende Bedingung erfüllt oder nicht erfüllt ist. Die Wahrscheinlichkeiten der Regeln, innerhalb derer die untersuchte Bedingung als indifferent ausgewiesen ist, bleiben unberücksichtigt.

Der *Suchaufwand* ergibt sich als Differenz aus Prüfaufwand und Beweissaufwand. Er entspricht dem mit der Summe der Wahrscheinlichkeiten derjenigen Regeln gewichteten Prüfaufwand, innerhalb derer die untersuchte Bedingung als indifferent ausgewiesen ist.

Beweisaufwand und Suchaufwand ergeben sich dann für die Beispieltabelle wie folgt:

		A_i	W_i^b	R1	R2	R3	R4	R5	Beweis-	Such-
				W_j^r 0,3	0,3	0,2	0,1	0,1	aufwand	aufwand
B1	Bedingung a	10	0,5	J	N	-	J	N	8	2
B2	Bedingung b	10	0,5	J	-	N	N	J	7	3
B3	Bedingung c	20	0,5	-	J	N	J	N	14	6

Entsteht nun aus der Tabellenübersetzung ein Programm, das die Bedingung a zuerst prüft, dann verändert sich der Suchaufwand der anderen beiden Bedingungen nicht, wie folgendes Schaubild zeigt:



		A_i	W_i^b	R1	R2	R3	R4	R5
				W_j^r 0,3	0,3	0,2	0,1	0,1
B1	Bedingung a	10	0,5	J	N	-	J	N
B2	Bedingung b	10	0,5	J	-	N	N	J
B3	Bedingung c	20	0,5	-	J	N	J	N

	A_i	W_i^b	R1 0,3	R3 0,1	R4 0,1	Such-aufwand
B2	10	0,5	J	N	N	0
B3	20	0,5	-	N	J	6

	A_i	W_i^b	R2 0,3	R3 0,1	R5 0,1	Such-aufwand
B2	10	0,5	-	N	J	3
B3	20	0,5	J	N	N	0

Suchaufwand für beide Teiltabellen
0 + 3 = 3
6 + 0 = 6

Abb. 20: Reinwald/Soland -Algorithmus: Ergebnis nach Prüfung der ersten Bedingung

Der bei den beiden Teiltabellen ausgewiesene Suchaufwand kann nun zur Entscheidung darüber herangezogen werden, welche Bedingung als nächste geprüft werden soll. Im Fall der den Ja-Zweig bildenden Teiltabelle ist dies die Bedingung b, bei der anderen Teiltabelle die Bedingung c. Danach wird noch die Prüfung der untersten Bedingung erforderlich, so dass folgende Entscheidungsbaumstruktur entsteht:

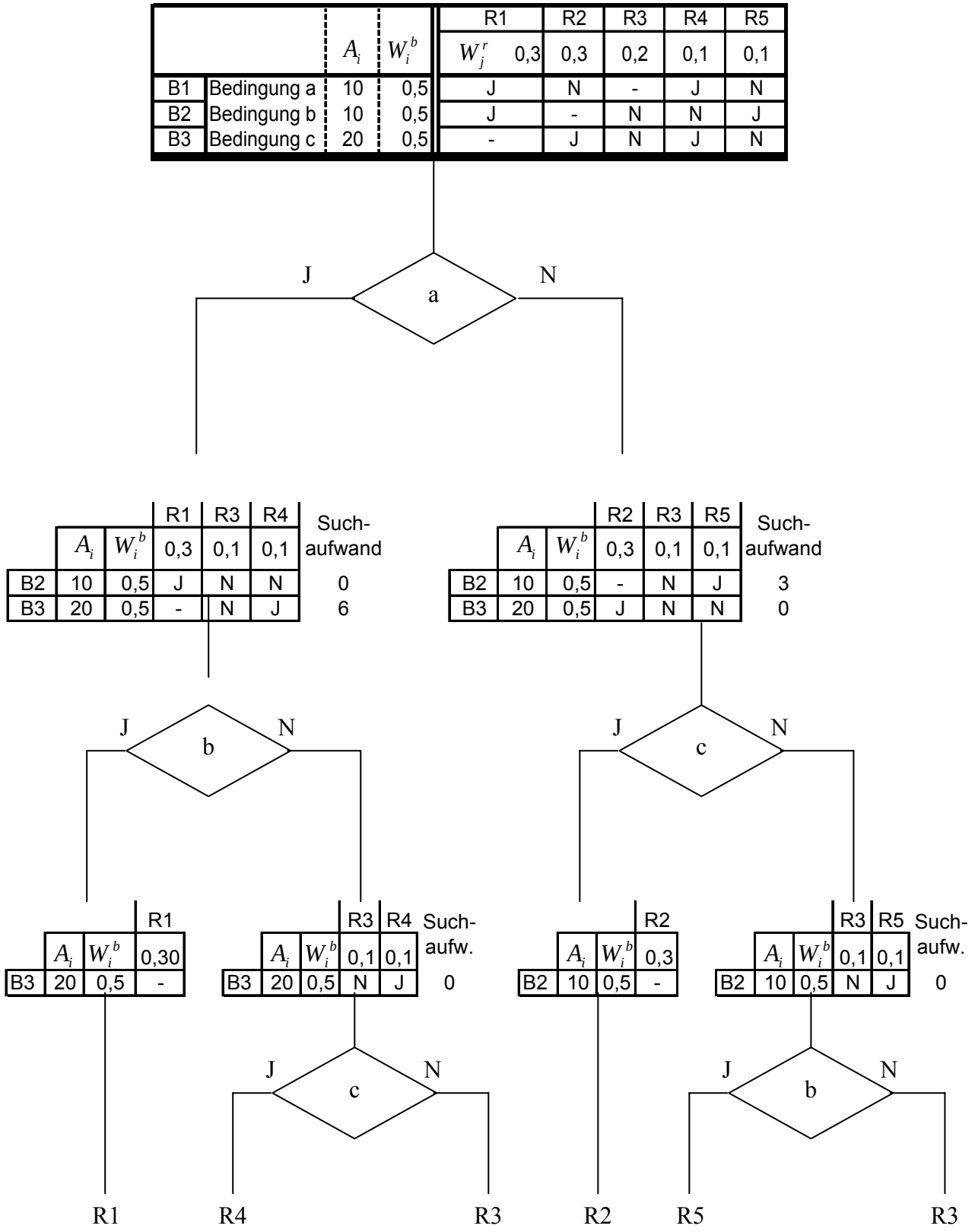

B1 Bedingung a 10 0,5 J N - J N
B2 Bedingung b 10 0,5 J - N N J
B3 Bedingung c 20 0,5 - J N J N
R1 R2 R3 R4 R5
W_j^r 0,3 0,3 0,2 0,1 0,1
a
J
N
B2 10 0,5 J N N 0
B3 20 0,5 - N J 6
B2 10 0,5 - N J 3
B3 20 0,5 J N N 0
Such-aufwand
b
c
B3 20 0,5 - R1
B3 20 0,5 N J R4 R3 0
B2 10 0,5 - R2
B2 10 0,5 N J R5 R3 0
Such-aufw.
R1
R4
R3
R2
R5
R3

Abb. 21: Reinwald/Soland-Algorithmus: Vollständige Zergliederung

Ein Programm, das nach dieser Struktur aufgebaut wird, weist einen Suchaufwand von 2 auf. Ergänzt zum Beweisaufwand in Höhe von 29 ergibt sich insgesamt ein Aufwand von 31.

Ein Suchaufwand von 2 gilt als Minimum für jedes Programm, das mit der Prüfung der Bedingung a beginnt. Jedes Programm, das zuerst die Bedingung b prüft, hat einen minimalen Suchaufwand von 3. Der minimale Suchaufwand beträgt 6 bei jedem Programm, das zuerst die Bedingung c prüft.

Nur dann, wenn bei der folgenden Aufteilung der Entscheidungstabelle kein Indifferenzanzeiger "zerlegt" werden muss, bleibt es bei dem ursprünglich bestimmten minimalen Suchaufwand. Die Beispieltabelle liefert dafür den Beweis. Weniger trivial aufgebaute Entscheidungstabellen erfordern einen höheren Suchaufwand. Dieser kann jedoch minimiert werden, wenn - wie demonstriert worden ist - jeweils als nächste zu prüfende Bedingung diejenige ausgewählt wird, die den geringsten Suchaufwand verursacht.

Die generelle Anwendbarkeit des Algorithmus von Reinwald und Soland muss ebenso wie die generelle Anwendbarkeit anderer optimierender Entscheidungstabellenalgorithmen verneint werden. Alle optimierenden Entscheidungsbaumalgorithmen setzen die logische Unabhängigkeit von Bedingungen voraus, da sie zur Erzielung ihres Optimierungseffekts in vielen Fällen die in der Tabelle vorzufindende Bedingungsreihenfolge verändern. In den Erörterungen zur Semantik der Entscheidungstabelle ist aber begründet worden, weshalb die Beibehaltung der Bedingungsreihenfolge bei logischer Abhängigkeit der Bedingungen ein semantisches Erfordernis sein kann. Optimierende Entscheidungsbaumalgorithmen bieten in diesen Fällen keine absolute Sicherheit einer korrekten Übersetzung.

Ein weiterer Nachteil der Entscheidungsbaumalgorithmen ist, dass sie keine Möglichkeit der direkten Übersetzung von Mehrtreffer–Entscheidungstabellen bieten. Solche Entscheidungstabellen müssten zunächst intern in Eintreffer-Entscheidungstabellen umgewandelt werden, bevor sie in eine Entscheidungsbaumstruktur umgewandelt werden können.

5.1.3.2.2.2 Maskenverfahren

Als wesentlichster Nachteil der manuellen Codiermethode "Regel für Regel" gilt im allgemeinen, dass jeder von "-" oder "#" verschiedene Bedingungsanzeiger zur Codierung der zugeordneten Bedingung führt. Auch zur Ausführungszeit ist die mehrfache Prüfung der Bedingungen erforderlich, wenn nicht die erste Regel bereits zutrifft.

Das zuerst von Kirk [1)] veröffentlichte Maskenverfahren behält die Regel-für-Regel-Prüfmethode bei, vermeidet jedoch die mehrfache Bedingungscodierung und -prüfung. Möglich wird das durch die Verwendung von Bitmatrizen, in denen die ursprüngliche Logik der Entscheidungstabelle zur Übersetzungszeit gespeichert wird, und durch die Umsetzung der Ergebnisse der zur Laufzeit ausgeführten Bedingungsprüfungen in einen Bitvektor, der dann mit den Spalten der Matrix so lange verglichen wird, bis eine zutreffende Spalte und damit die passende Regel gefunden wurde. Ihre Bezeichnung haben die Maskenverfahren von der "masking matrix" erhalten, die Kirk einführen musste, da er zur Verschlüsselung der Bedingungsanzeiger nur ein Bit benutzte, mit "Y", "N" und "-" jedoch drei Anzeigerformen zu verschlüsseln hatte.[2)]

Offensichtlich unabhängig von Kirk entwickelte auch Press ein Maskenverfahren, das sich von Kirks Vorschlag davon unterschied, dass 2 Bits zur Verschlüsselung der Anzeiger bereitgestellt wurden, so dass keine Notwendigkeit der Definition einer Maskenmatrix bestand.[3)]

Die Maskenmethode wurde nach Kirks und Press' 1965 erfolgten Veröffentlichungen von anderen Autoren mehrfach verbessert und in ihrer Anwendungsmöglichkeit verbreitert.

King [4)] kritisierte die Notwendigkeit, in jedem Fall alle Bedingungen vor der Auswertung der Matrix zu prüfen und schlug als Modifikation die "interrupted rule mask procedure" vor, die unter Verwendung von relativen Häufigkeiten der Regelausführung sowie der Zeiten, die zur Bedingungsprüfung benötigt werden, günstigere Laufzeitergebnisse bringt.[5)]

Björk [6)] war offensichtlich der erste, der ein Verfahren fand, die Bitmatrix zeilenweise auszuwerten, und deshalb immer dann auf die Prüfung weiterer Bedingungen verzichten konnte, wenn nach Prüfung der letzten Bedingung die zutreffende Regel bereits feststand. Außerdem gab er auch die ersten Hinweise zur direkten Übersetzung erweiterter Entscheidungstabellen mit dem Maskenverfahren.[7)]

---

1) Vgl. Kirk (Use of decision tables)
2) Vgl. Kirk (Use of decision tables), S. 42
3) Vgl. Press (Conversion of decision tables), S. 388
4) Vgl. King (Conversion of decision tables)
5) Vgl. King (Conversion of decision tables), S. 797 ff.
6) Vgl. Björk (Decision tables in ALGOL 60)
7) Vgl. Björk (Decision tables in ALGOL 60), S. 150

Barnard [1]gebührt der Verdienst, die Maskentechnik auf die Mehrtreffer-Entscheidungstabelle ausgedehnt zu haben, wobei er die Mehrtreffer-Entscheidungstabelle bei dieser Gelegenheit erst entdeckte.

Muthukrishnan und Rajaraman [2] schließlich legten 1970 ein Verfahren vor, das die Entscheidungstabelle, wie von Press vorgeschlagen, in eine Bitmatrix verschlüsselte. Sie fanden jedoch ein eleganteres Verfahren der Regelauswahl, das auf einer zeilenweisen Auswertung der Matrix basiert.[3] Sie entwickelten einen zweiten Algorithmus zur direkten Verarbeitung erweiterter Entscheidungstabellen,[4] der ein ausgereifteres Bild bietet als der Vorschlag Björks, und sie wiesen schließlich darauf hin, dass bei dem von ihnen vorgeschlagenen Verfahren die Analyse von Mehrdeutigkeiten auch zur Laufzeit erfolgen kann.[5]

Entscheidungstabellenalgorithmen nach dem Maskenverfahren sind heute so ausgereift, dass sie alle Anforderungen an die Entscheidungstabellenverarbeitung erfüllen. Sie können so implementiert werden, dass sie die Bedingungen in der angegebenen Reihenfolge überprüfen, sie erlauben eine direkte Übersetzung erweiterter Tabellen, sie verarbeiten Mehrtreffer-Entscheidungstabellen, sie können den Ausschlußanzeiger bewältigen, sich überschneidende Bedingungsanzeiger werden richtig ausgewertet und Eindeutigkeits- und Vollständigkeitsprüfung sind auch zur Ausführungszeit möglich.[6] Sie haben nur einen Nachteil: Die Verarbeitungszeiten der generierten Programme sind ungünstiger als bei Verwendung eines Entscheidungsbaumverfahrens.

Das stellt Entscheidungstabellenanwender und Implementatoren von Entscheidungstabellenübersetzern vor ein Dilemma. Entscheiden sie sich für einen Übersetzer mit Entscheidungsbaumalgorithmus, können sie zwar mit einem laufzeitgünstigen Code rechnen, für die Erstellung der automatisch zu verarbeitenden Entscheidungstabellen müssen jedoch Restriktionen hingenommen werden, die einen erheblichen Teil der Flexibilität der Entscheidungstabelle wieder aufheben.·

---

1) Vgl. Barnard (New rule mask technique)
2) Vgl. Muthukrishnan; Rajaraman (Conversion of decision tables)
3) Vgl. Muthukrishnan; Rajaraman (Conversion of decision tables), S. 348
4) Vgl. Muthukrishnan; Rajaraman (Conversion of decision tables), S. 348 f.
5) Vgl. Muthukrishnan; Rajaraman (Conversion of decision tables), S. 350 f.
6) Ein Algorithmus, der alle diese Fähigkeiten in sich vereinigt, ist im Entscheidungstabellenvorübersetzer VORELLE implementiert.
Vgl. mbp (VORELLE-Handbuch)

Entscheiden sie sich dagegen für einen Übersetzer mit Maskenalgorithmus, entfallen alle Restriktionen im Hinblick auf die Verarbeitbarkeit von Entscheidungstabellen, alle in der Entscheidungstabellentheorie entwickelten Entscheidungstabellenfähigkeiten werden dem Anwender nutzbar gemacht. Die entstehenden Programme benötigen jedoch mehr Verarbeitungszeit als Programme, die von einem Entscheidungsbaumverfahren generiert wurden.

Zur Lösung dieses Dilemmas bieten sich zwei Wege an. Zum einen kann damit gerechnet werden, dass sowohl Entscheidungsbaumverfahren als auch Maskenverfahren in der Zukunft weiterentwickelt werden. Durch die Ergebnisse der Weiterentwicklungen mag das erläuterte Dilemma entschärft werden. Zum anderen ist vorstellbar, dass in einem Entscheidungstabellenvorübersetzer beide Algorithmentypen implementiert sind. Dabei sollte vorwiegend mit dem Maskenalgorithmus gearbeitet werden. Nur Entscheidungsstabellen, die an einer sehr aktiven Stelle des Programms stehen, werden mit dem Entscheidungsbaumverfahren übersetzt, nachdem sie vorher gegebenenfalls in eine Form transformiert worden sind, die mit diesem Algorithmentyp verarbeitbar ist.

5.1.4 Der Beitrag der Entscheidungstabellentechnik zur Lösung des Problems der Verifikation von Programmen

Die lange Zeit bei Datenverarbeitungsanwendern geübte Praxis, ein Programm nach seiner Übersetzung und nach der Beseitigung von Syntaxfehlern einem empirischen Ablauftest zu unterziehen und es bei positivem Ergebnis für einsetzbar zu erklären, wird in den letzten Jahren immer stärker angezweifelt. Treffend beschreibt ein einziger - bereits an anderer Stelle zitierter - Satz Dijkstras das Problem:

"Program testing can be used to show the presence of bugs but never to show their absence!" [1)]

Zahlreiche Veröffentlichungen der letzten Zeit machen Vorschläge, wie ein Programm *verifiziert* werden kann. Im Folgenden soll versucht werden, die Stellung der Entscheidungstabellentechnik im Rahmen der Programmverifikation zu klären.

---

1) Vgl. Dijkstra (Structured programming), S. 85

5.1.4.1 Stufen des Verifikationsprozesses

Die Verifikation eines Programms beginnt bereits in der Entwurfsphase. Schon in der Phase des organisatorischen Systementwurfs muss eine Begründung für jede definierte *Informationsaufgabe* gefunden werden.

Verifikationsbedürftig ist auch die *Methode der Aufgabenerfüllung*, die in der Regel ebenfalls in der Phase des organisatorischen Systementwurfs entwickelt wird.

Der *Algorithmus*, der entworfen wird, um die Aufgabenerfüllung zu automatisieren, kann nach zwei Gesichtspunkten verifiziert werden. Eine Prüfung der *Struktur* des Algorithmus muss Aufschluss über strukturelle Konsistenz und Vollständigkeit geben. Eine Prüfung des logischen *Inhalts* des Algorithmus muss seine Fähigkeit nachweisen, alle möglichen Eingabedaten korrekt zu verarbeiten.

Schließlich verbleibt als Objekt der Verifikation der Programmtext. Der *Programmtext* ist in einer formalen Sprache abgefasst, die über klar definierte Syntaxregeln verfügt. Verifiziert werden muss dann, dass alle Bestandteile des Programmtextes diesen Syntaxregeln entsprechend abgefasst sind.

Somit lässt sich die Verifikation als ein mehrstufiger Prozess verstehen, der sich auf fünf unterschiedliche Abstraktionsebenen des Informationssystems erstreckt.

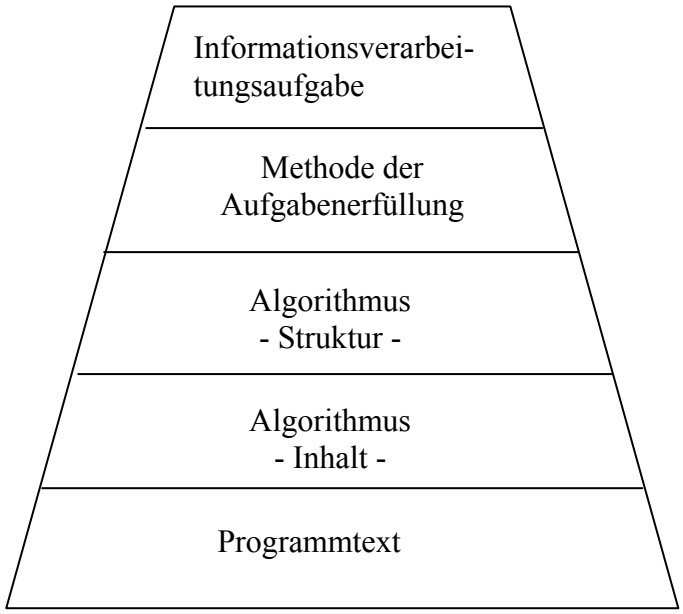


Abb. 22: Modell der Abstraktionsebenen des Informationssystems

Jede dieser fünf Abstraktionsebenen erlaubt eine von den anderen Ebenen unabhängige Verifizierung. Zu beachten bleibt jedoch, dass immer dasselbe Objekt verifiziert wird: der untersuchte Teil eines rechnergestützten Informationssystems.

Fehler können, müssen sich aber nicht auf mehrere der definierten Abstraktionsebenen auswirken. So kann die Verifikation des Programmtextes ergeben, dass keine Syntaxfehler vorhanden sind. Dennoch kann der Algorithmus Fehler aufweisen. Der Algorithmus wiederum kann sowohl strukturell als auch inhaltlich die ausgearbeitete Aufgabenerfüllungsmethode korrekt in sequentiell ausführbare automatische Datenverarbeitungsschritte überführen. Die Methode der Aufgabenerfüllung kann jedoch ungeeignet sein, das durch die Aufgabe gegebene Datenverarbeitungsproblem zu lösen.

Ein logisch inkonsistenter Algorithmus kann aber auf eine fehlerhafte Aufgabenerfüllungsmethode zurückgehen oder ein inhaltlich unvollständiger Algorithmus an strukturellen Unvollständigkeiten erkennbar sein.

5.1.4.2 Verifikationsmethoden und –techniken

Die Verifikation des Programmtextes ist heute ein vollständig automatisierter Vorgang. Assemblierer oder Kompilierer unterziehen den eingegebenen Programmtext einer ausführlichen Syntaxanalyse, die die Einhaltung der syntaktischen Bildungsregeln für elementare und zusammengesetzte syntaktische Einheiten überprüft. Abweichungen von der zulässigen Syntax werden in Form von Fehlermeldungen ausgegeben. Wenn auch die Fähigkeiten zur Erkennung von Syntaxfehlern bei den verfügbaren Assemblierern und Kompilierern noch verbesserungsbedürftig sind, [1)] so kann die Verifikation des Programmtextes heute nicht mehr als Engpass einer wirksamen Programmerstellung gelten.

Die inhaltliche Verifikation eines Algorithmus kann über einen empirischen Ablauftest erfolgen. Vollständig verifiziert wird ein Algorithmus mit dieser Methode jedoch nur dann, wenn alle theoretisch möglichen Verarbeitungsfälle mit Hilfe entsprechender Eingabedaten getestet werden, das heißt, die Verarbeitungsergebnisse aller durch diese Eingabedaten beschriebenen Fälle mit Hilfe anderer Methoden bestimmt werden und ein Vergleich mit den automatisch erstellten Ergebnisdaten erfolgt. Die Schwächen des empirischen Ablauftests liegen darin begründet, dass die

---

1) Vgl. Schmitz; Seibt; u.a. (Modell der Wirksamkeit), S. 92

theoretisch mögliche Anzahl von unterschiedlich behandelten Verarbeitungsfällen bereits bei relativ kleinen Programmen so groß sein kann, dass eine empirische Verifikation des Algorithmus weder zumutbar noch wirtschaftlich ist. Enthält ein Algorithmus nur 10 Bedingungen mit jeweils 2 Zuständen, die jeden Verarbeitungsfall beeinflussen, so wird bereits die Untersuchung von $2^{10} = 1024$ Testfällen notwendig, um einen vollständigen empirischen Ablauftest durchzuführen.[1)] Stichprobenanalysen nach gesicherten statistischen Methoden sind nicht möglich, da das Gesetz der großen Zahlen in der Programmierung nicht anwendbar ist.[2)]

Gleichwohl gilt der empirische Ablauftest in der Praxis als bedeutendstes Verfahren der inhaltlichen Verifikation von Algorithmen. Mehr oder weniger intuitiv wird eine Auswahl "repräsentativer" Verarbeitungsfälle definiert und der Algorithmus damit überprüft. Das Programm gilt als verifiziert, wenn es diese Testfälle korrekt verarbeitet.

Da die Erstellung von Testdaten, die Durchführung der Vergleichsrechnungen und der Vergleich manuell errechneter mit automatisch abgeleiteten Ergebnisdaten selbst wiederum ein fehleranfälliger Prozess ist, sind verschiedene automatische Hilfsmittel entwickelt worden, die den Rechner in diesen Vorgang so weit wie möglich einschalten. Hier sind Testdatengeneratoren zu erwähnen, die aufgrund einer manuell zu erstellenden Spezifikation Datensätze generieren, deren Inhalt nach dem Zufallsprinzip automatisch erstellt wird. Es verbleibt dann jedoch noch die Aufgabe, die Richtigkeit der Verarbeitungsergebnisse nachzuprüfen.[3)] Auch Simulationssysteme, die die Umgebung eines Programms simulieren, um es überhaupt einem empirischen Ablauftest unterziehen zu können, sind in der Praxis eingesetzt. Beispielsweise simulieren sie einen noch nicht vorhandenen Steuermodul, wenn ein Verarbeitungsmodul getestet werden soll.[4)] Alle Testhilfsmittel und Teststrategien [5)] können jedoch das Grundproblem des empirischen Ablauftests nicht aus der Welt schaffen: die nicht gesicherte Vollständigkeit.

---

1) Interessant ist auch der bei Boehm zu findende relativ kleine Programmablaufplan, der allerdings 2 Programmzyklen enthält. Boehm beziffert die Anzahl der logischen Wege durch dieses Programm mit 10^{20}. Vgl. Boehm (Software and its impact), S. 58
2) Vgl. Hetzel (Program testing), S. 22 f.
3) Vgl. Goodenough; Ross; Eanes (MAIDS), S. 91 f.
4) Vgl. Hetzel (Program testing), S. 21
5) Beispielsweise Integrationstest, Regressionstest, Top-down-Test, Bottom-up-Test. Vgl. Goodenough; Ross; Eanes (MAIDS), S. 85 ff.

Einen anderen Weg der Verifikation von Algorithmen gehen alle Verfahren, die die Richtigkeit eines Programms mit mathematischen Methoden beweisen wollen. Obgleich die Theorie des Programmbeweises noch offene Probleme aufweist, existieren mittlerweile zahlreiche Beweistechniken, deren Praktikabilität bereits nachgewiesen wurde.[1)] Der Beweis eines Programms ist definiert worden als Demonstration der Konsistenz zwischen einem Programm und den Behauptungen über die Leistungen, die dieses Programm erbringen soll.[2)] Diese Demonstration wird mit Hilfe streng mathematischer Verfahren erbracht. Das bekannteste Beweisverfahren ist die auf Floyd [3)] zurückgehende Invariantenmethode. Behauptungen über Beziehungen zwischen den Variablen eines Programms werden so zwischen den Programmtext eingeschoben, dass jeder Zyklus mindestens von einer solchen Behauptung unterbrochen wird. Jede Behauptung beansprucht für sich die Eigenschaft, dass sie jedes Mal gilt, wenn die Programmsteuerung ihre Stelle im Programm erreicht hat, sie verhält sich also invariant. Aufgabe eines nach mathematischen Regeln durchzuführenden Beweises ist nun der Nachweis, dass jede Behauptung wahr ist.[4)]

Umfangreiche Forschungs- und Entwicklungsarbeiten werden derzeit durchgeführt, um den Aufwand zur Durchführung eines Programmbeweises unter Einsatz eines Rechners auf ein wirtschaftlich vertretbares Niveau zu drücken.[5)]

Es ist jedoch noch nicht abzusehen, wann die Datenverarbeitungspraxis diese Beweistechniken auf breiter Basis einsetzen kann.

Die *strukturelle* Verifikation eines Algorithmus ist verhältnismäßig einfach durchführbar, wenn der Algorithmus unter Verwendung einer formalen Strukturspra che formuliert worden ist. Die Verifikation besteht dann in dem Nachweis, dass die Syntaxregeln der Strukturspra che eingehalten worden sind.

---

1) Vgl. dazu die von London erarbeitete Zusammenstellung in London (Proving programs correct), S. 41 ff.
2) Vgl. London (Program verification), S. 1
3) Vgl. Floyd (Assigning meanings to programs), zitiert nach London , (Program verification), S. 3
4) Vgl. London (Program verification), S. 3.
5) Zu einem interaktiven System zum rechnergestützten Beweis von PASCAL-Programmen vgl. Good; London; Bledsoe (Interactive program verification); Igarashi; London; Luckham (Automatic program verification)

Eine häufig angewendete Struktursprache ist beispielsweise der Programmablaufplan. Eine Syntaxregel zum Aufbau von Programmablaufplänen ist, dass ein Verzweigungssymbol mindestens zwei Programmzweige initiieren muss, da andernfalls die beschriebene Bedingung den Programmablauf nicht beeinflussen kann. Die Verifikation dieser Syntaxregel ist jedem Programmierer möglich.

Auch die strukturierte Programmierung hat in Bezug auf ihre Vorschrift zur Anwendung der elementaren Ablaufstrukturen den Charakter einer Struktursprache. Die strukturelle Verifikation eines "strukturierten" Programms besteht in dem Nachweis, dass das gesamte Programm aus den zugelassenen elementaren Ablaufstrukturen aufgebaut ist.

Die Entscheidungstabellentechnik als außerordentlich weit entwickelte Struktursprache erlaubt eine strukturelle Verifikation aller Bestandteile des Algorithmus, die in Form von Entscheidungstabellen formuliert sind. Die wichtigsten Verifikationstechniken sind Eindeutigkeitsprüfung und Vollständigkeitsprüfung.

Zur Verifikation der *Aufgabenerfüllungsmethode* sind - außer möglicherweise anwendbaren mathematischen Beweisverfahren - keine allgemeingültigen Techniken außer empirischen Verfahren bekannt.

Die Verifikation der zugrundeliegenden Informationsaufgabe schließlich wird ebenfalls nur empirisch durchgeführt werden können.

5.1.4.3 Entscheidungstabellentechnik als Verifikationstechnik

Die Entscheidungstabellentechnik in einer rechnergestützten Anwendung hat als Verifikationstechnik eine größere pragmatische Bedeutung, als die Ausführungen im vorigen Abschnitt zunächst vermuten lassen.

Die strukturelle Verifikation eines Algorithmus mit Hilfe der in der Entscheidungstabellentechnik entwickelten Verfahren kann bereits eine erhebliche Anzahl inhaltlicher Fehler des Algorithmus aufdecken. Anwender der Entscheidungstabellentechnik berichten, dass nach Beseitigung aller Redundanzen, Widersprüche und Unvollständigkeiten in entscheidungstabellenintensiven Programmen kaum noch logische Fehler während des empirischen Ablauftests gefunden werden.[1)]

---

1) Diese Auskunft erhielt der Verfasser mehrfach von Anwendern einer rechnergestützten Entscheidungstabellentechnik

Entscheidungstabellen erlauben darüber hinaus auch noch eine wirksamere Ausführung empirischer Ablauftests. Die Erstellung von Testdaten kann sich an den Entscheidungsregeln der im Programm enthaltenen Entscheidungstabellen, die jeden unterschiedlichen Verarbeitungsfall klar definieren, orientieren. Die Testausführung lässt sich bei einer mit allen modernen Entscheidungstabellenübersetzern möglichen automatischen Regelhäufigkeitsstatistik protokollieren. Damit lässt sich sehr einfach nachweisen, ob alle Verarbeitungsalternativen in einem Programm mindestens einmal getestet worden sind.

Das Ausmaß, in der sich die Entscheidungstabellentechnik bei der Verifikation eines Programms insgesamt positiv auswirkt, ist jedoch vom Anteil der Entscheidungstabellen im Programm und von der Art der Algorithmenbestandteile außerhalb der Entscheidungstabellen abhängig. Wenn ein Algorithmus wenig Entscheidungslogik enthält, müssen andere Verifikationstechniken angewendet werden.

5.1.5 Programmoptimierung bei Einsatz von Entscheidungstabellen

Der Optimierung [1)] von Programmen kommt in der Systemgestaltung immer dann besondere Bedeutung zu, wenn sich Engpässe bei den zur Informationsverarbeitung eingesetzten Ressourcen zeigen oder solche Engpässe während der Systementwurfsphase bereits antizipiert werden können.

Das zur Erläuterung der Verifikationsmöglichkeiten benutzte Abstraktionsebenenmodell eines Informationssystems lässt sich mit einer geringen Modifikation auch zur Einordnung von Optimierungsmöglichkeiten benutzen. Als oberste Ebene wird lediglich der Aktionsträger ergänzt.

---

1) Da im Zeitalter der virtuellen Speichertechnik und immer billigerer Hauptspeicher die Hauptspeicherminimierung von Programmen wesentlich weniger Bedeutung hat als die Laufzeitminimierung, wird im folgenden davon ausgegangen, dass die Laufzeitminimierung als Optimierungsziel gilt.

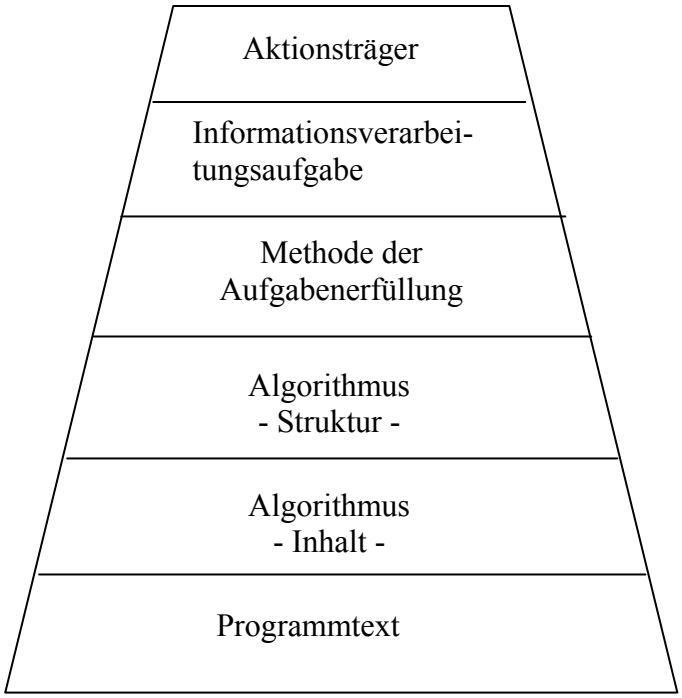


Abb. 23: Erweitertes Modell der Abstraktionsebenen des Informationssystems

Auf der Ebene des Programmtextes wird optimiert, wenn bekannt laufzeitaufwendige Sprachelemente der benutzten Programmiersprache durch andere, laufzeitgünstigere Formulierungen ersetzt werden.

Ein Algorithmus wird inhaltlich optimiert, wenn vorgesehene Datenverarbeitungsschritte durch andere, weniger aufwendige ersetzt werden können.

Ein Algorithmus wird strukturell optimiert, wenn es gelingt, durch strukturelle Umformungen die Anzahl der ausgeführten Arbeitsschritte zu minimieren, dabei aber keine inhaltliche Änderung der Verarbeitung erfolgt.

Die Methode der Aufgabenerfüllung wird optimiert, wenn sie so umgestaltet werden kann, dass auf der Basis der neuen Methode ein weniger aufwendiger Algorithmus erstellt werden kann.

Die Informationsverarbeitungsaufgabe kann beispielsweise durch Reduzierung der Anforderungen an die Ausgabeinformationen optimiert werden. Es kann gelingen, einen Teil der gewünschten Ausgabeinformationen im Leistungsumfang anderer Informationsaufgaben zu lokalisieren, so dass statt eigener Verarbeitungsprozesse nur Übermittlungsvorgänge erforderlich werden.

Auf der Ebene des Aktionsträgers schließlich wird optimiert, wenn die Informationsverarbeitungskapazität des Aktionsträgers durch Beseitigung von Engpässen vergrößert wird.

Der Einflussbereich der Entscheidungstabellentechnik erstreckt sich im Rahmen der Programmoptimierung auf die Ebenen der strukturellen und inhaltlichen Merkmale des Algorithmus und des Programmtextes.

Wie in dieser Arbeit ausführlich gezeigt wurde, wirkt sich die Minimierung von Entscheidungstabellen so aus, dass bestimmte Bedingungen innerhalb der Verarbeitungsregeln als indifferent ausgewiesen werden, so dass bei Zutreffen dieser Regeln auf ihre Prüfung verzichtet werden kann.[1)] Dies kann als strukturelle Optimierung gelten. Auch eine erfolgreiche Zergliederung einer Entscheidungstabelle durch kombinierte Bedingungs-/Aktionsaufteilung [2)] lässt sich als strukturelle Optimierung kennzeichnen, da dadurch ebenfalls Bedingungen aus den Abläufen eliminiert werden, die sich bei bestimmten Bedingungskonstellationen in Bezug auf die Auswahl der Aktionsfolgen indifferent verhalten. Durch Kombination von Entscheidungstabellen kann möglicherweise die mehrfache Prüfung derselben Bedingung verhindert werden. Auch das lässt sich als strukturelle Optimierung bezeichnen.

Bei der rechnergestützten Anwendung der Entscheidungstabellentechnik verhilft die Testhilfe "Regelhäufigkeitsstatistik" zu wertvollen Informationen, die zu einer Optimierung inhaltlicher Aspekte des Algorithmus und der bei der Codierung benutzten Sprachelemente benutzt werden können. Bei den ersten Praxisläufen aktiviert und ausgewertet, zeigt diese Testhilfe genau, welche Entscheidungstabellen am häufigsten durchlaufen worden sind, und innerhalb dieser Entscheidungstabellen wiederum werden die aktivsten Regeln ausgewiesen. So wird eine gezielte Optimierung von Bedingungsprüfungen und Aktionsfolgen ermöglicht und die Aufmerksamkeit des Programmierers auf wenige Anweisungen gerichtet, deren Ersetzung durch weniger aufwendige Formulierungen einen unverhältnismäßig großen Optimierungseffekt bringen kann.

---

1) Vgl. dazu ·Abschnitt 1.1.2.1.2.1.2

2) Vgl. dazu Abschnitt 1.1.2.2.1

5.1.6 Fallbeispiel zum Einsatz eines Entscheidungstabellen -Vorübersetzers

Als anschauliches Beispiel für die Einsatzmöglichkeiten eines Entscheidungstabellen-Vorübersetzers soll ein Fallbeispiel erläutert werden, das mit Hilfe von VORELLE, dem Entscheidungstabellen-Vorübersetzer des mbp, Mathematischer Beratungs- und Programmierungsdienst GmbH, bearbeitet wurde. Die Problemstellung wurde in Anlehnung an ein Beispiel formuliert, das von Verhelst in die Entscheidungsstabellen-Literatur eingeführt wurde.[1)]

Kernstück der Problemstellung ist die Berechnung einer Jahresprämie, die sich aus mehreren Bestandteilen zusammensetzt und in Abhängigkeit von der Betriebszugehörigkeit des Arbeitnehmers, der Anzahl Fehltage im abgelaufenen Jahr und einer individuellen Produktivitätsziffer berechnet wird. Es sei angenommen, dass die erste Analyse des Problems zu folgender Entscheidungstabelle geführt hat:

Hauptprämie		1	2	3	4	5	6	7	8	9	10	11
B1	Betriebszugehörigkeit	-	>2	>2	>2	>2	1-2	1-2	1-2	1-2	Ø	Ø
B2	Fehltage	Ø-1	Ø-1	2-9	10-19	10-19	Ø-1	2-9	Ø-1	2-9	Ø-1	Ø-1
B3	Produktivitätszahl N>15	J	N	-	J	-	J	J	N	N	-	J
A1	Dienstalterprämie = Gehalt * 15/100	X	X	X	X	X	-	-	-	-	-	-
A2	Anwesenheitsprämie -1 = Gehalt * 20/100 - Fehltage * Gehalt * 2/100	X	X	X	-	-	X	X	X	X	-	-
A3	Anwesenheitsprämie -2 = Gehalt * 5/100	-	-	-	-	-	-	-	-	-	X	X
A4	Produktivitätsprämie -1 Gehalt * 20/100 *N/20	X	X	X	X	-	X	X	-	-	-	-
A5	Produktivitätsprämie -2 Gehalt * 10/100 *N/20	-	-	-	-	-	-	-	-	-	-	X
A6	Prämie für besondere Leistungen: Gehalt * 2/100 * N/20	X	-	-	-	-	X	-	-	-	-	X
A7	Gehe nach "Sonderprämie"	X	X	X	X	X	X	X	X	X	X	X

1) Vgl. Verhelst (Constructing decision tables), S. 27 ff.

Obwohl Bedingungen. Aktionen und Anzeiger noch nicht in einer formalen Sprache abgefasst sind lässt sich bereits die Struktur der Entscheidungstabelle verifizieren.

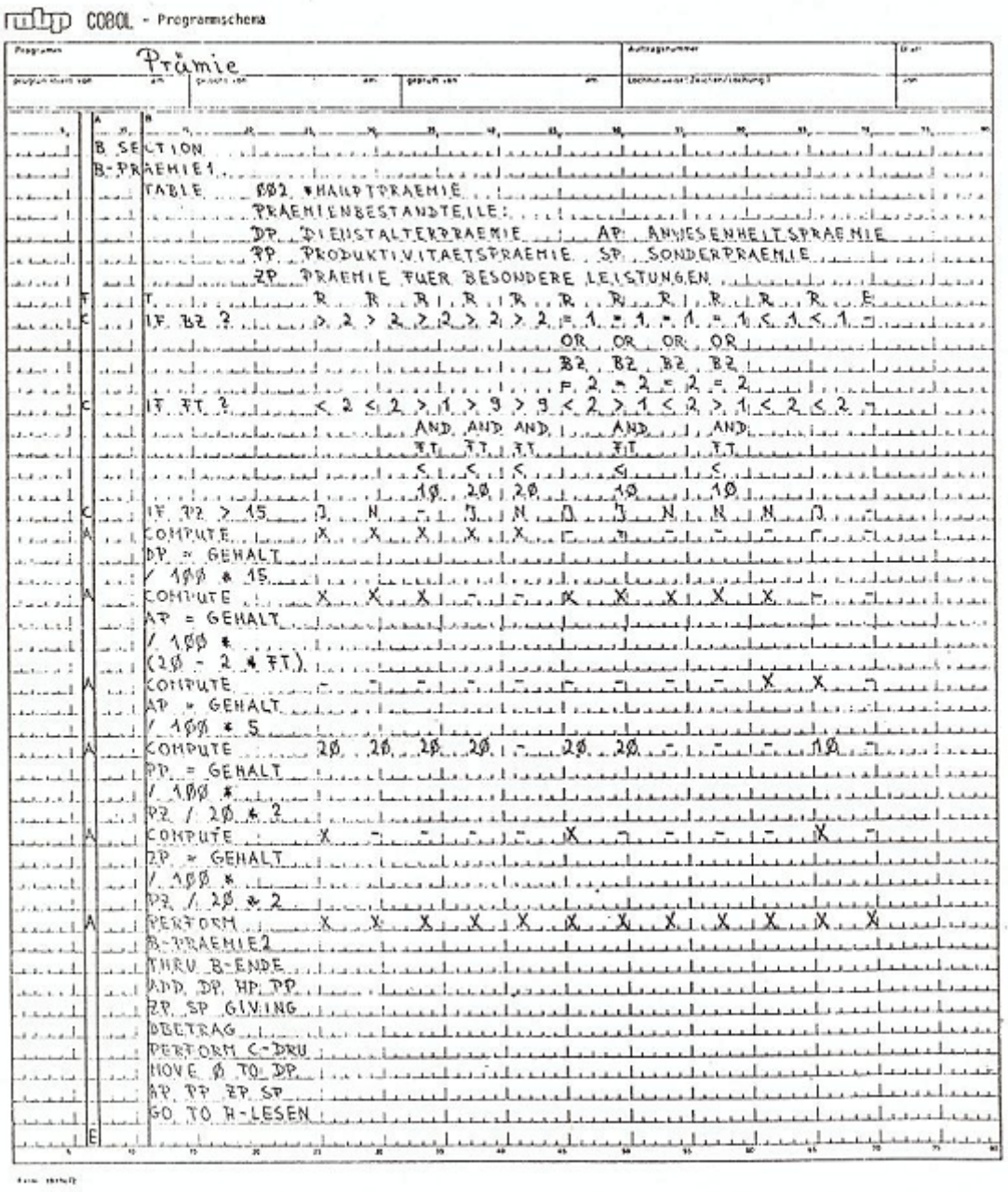
Nach Eingabe der Entscheidungstabelle in VORELLE wird unter der Bezeichnung "Logikanalyse" folgender Output produziert:

```
                                MBP - V O R E L L E          VERSION  COBOL-DOS

PROGRAM:PRAEVERB                      LOGIKANALYSE

TABLE    002

F    T                          R    R    R    R    R    R    R    R    R    R    R
                                0    0    0    0    0    0    0    0    0    1    1
                                1    2    3    4    5    6    7    8    9    0    1

C 01 BETRIEBSZUGEHOERIG-        -    MEHR MEHR MEHR MEHR 1    1    1    1    0    0
       KEIT 7 JAHRE                  ALS  ALS  ALS  ALS  BIS  BIS  BIS  BIS
                                     2    2    2    2    2    2    2    2
C 02 ANZAHL FEHLTAGE ?          0    0    2    10   10   0    2    0    2    0    0
                                ODER ODER BIS  BIS  BIS  ODER BIS  ODER BIS  ODER ODER
                                1    1    9    19   19   1    9    1    9    1    1
C 03 PRODUKTIVITAETSZAHL        J    N    -    J    -    J    J    N    N    -    J
       UEBER 10
-------------------------------------------------------------------------------------
A 01 DIENSTALTERPRAEMIE =       X    X    X    X    X    -    -    -    -    -    -
       GEHALT * 15 / 100
A 02 ANWESENHEITSPRAEMIE-1      X    X    X    -    -    X    X    X    X    -    -
       = GEHALT * 20 / 100
       - FEHLTAGE * GEHALT
       * 2 / 100
A 03 ANWESENHEITSPRAEMIE-2      -    -    -    -    -    -    -    -    -    X    X
       = GEHALT * 5 / 100
A 04 PRODUKTIVITAETSPRAE-       X    X    X    X    -    X    X    -    -    -    -
       MIE-1 = GEHALT * 20 /
       100 * PRODUKTIVI-
       TAETSZAHL / 20
A 05 PRODUKTIVITAETSPRAE-       -    -    -    -    -    -    -    -    -    -    X
       MIE-2 = GEHALT * 10 /
       100 * PRODUKTIVI-
       TAETSZAHL / 20
A 06 PRAEMIE FUER BESONDERE     X    -    -    -    -    X    -    -    -    -    X
       LEISTUNGEN: GEHALT *
       2 / 100 * PRODUKTIVI-
       TAETSZAHL / 20
A 07 GEHE NACH                  X    X    X    X    X    X    X    X    X    X    X
       'SONDERPRAEMIE'


TABLE    002   DIE ANALYSE BASIERT AUF FOLGENDEN BEDINGUNGSANZEIGERN:

C 01 I MEHR I 1   I 0 I
     I ALS  I BIS I   I
     I 2    I 2   I   I
-----------------------
C 02 I 0    I 2   I 10  I
     I ODER I BIS I BIS I
     I 1    I 9   I 19  I
-----------------------
C 03 I Y I N I
```

```
                                  HBP - V O R E L L E          VERSION  COBOL-DOS

PROGRAM:PRAEVERB                        LOGIKANALYSE

TABLE    002   ERGEBNIS DER VOLLSTAENDIGKEITSPRUEFUNG:
               DIE TABELLE IST UNVOLLSTAENDIG.
               ES FEHLEN FOLGENDE KONSTELLATIONEN:

     I 0   I 0   I 0   I
     I 0   I 0   I 0   I
     I 1   I 2   I 3   I
-----+-----+-----+-----+
C 01 I 1   I 0   I 0   I
     I BIS I     I     I
     I 2   I     I     I
-----+-----+-----+-----+
C 02 I 10  I 2   I 10  I
     I BIS I BIS I BIS I
     I 19  I 9   I 19  I
-----+-----+-----+-----+
C 03 I -   I -   I -   I
     I     I     I     I
-----+-----+-----+-----+


TABLE    002   ERGEBNIS DER REDUNDANZPRUEFUNG:
               DIE TABELLE ENTHAELT REDUNDANZEN UND/ODER WIDERSPRUECHE.

     IR IR IR I
     I0 I0 I1 I
     I1 I5 I0 I
-----+--+--+--+
R 01 I  I  IWUI
-----+--+--+--+
R 04 I  IWEI  I
-----+--+--+--+
R 06 IWEI  I  I
-----+--+--+--+
R 11 IWEI  IWEI
-----+--+--+--+

      R=REDUNDANZ  W=WIDERSPRUCH
      I=IDENTITAET U=UEBERSCHNEIDUNG E=SPALTENREGEL ENTHAELT ZEILENREGEL.


TABLE    002   GEMEINSAME BEDINGUNGSKONSTELLATIONEN BEI UEBERSCHNEIDUNGEN (WU ODER RU):

     I R R  I
     I 0 1  I
     I 1&0  I
-----+------+
C 01 I 0    I
     I      I
     I      I
-----+------+
C 02 I 0    I
     I ODER I
     I 1    I
-----+------+
C 03 I J    I
     I      I
-----+------+


TABLE    002   E N D E  DER ANALYSE
```

Obwohl alle wiedergegebenen Analyseergebnisse nur auf der Basis von Syntaxmerkmalen erarbeitet wurden, sind sie auch semantisch relevant, wenn - wie im Beispiel - die Bedingungsanzeiger logisch disjunkt sind.

Es sei nun angenommen, dass die enthaltenen Fehler korrigiert wurden und die Entscheidungstabelle nunmehr als Vorlage für die Codierung benutzt werden kann. Bei Anwendung von COBOL ergibt sich:


COBOL - Programmschema
Prämie
B SECTION.
B-PRAEMIE1.
TABLE 002 *HAUPTPRAEMIE
PRAEMIENBESTANDTEILE:
DP DIENSTALTERSPRAEMIE AP ANWESENHEITSPRAEMIE
PP PRODUKTIVITAETSPRAEMIE SP SONDERPRAEMIE
ZP PRAEMIE FUER BESONDERE LEISTUNGEN.
IF B2 ? > 2 > 2 > 2 > 2 > 2 = 1 = 1 = 1 = 1 < 1 < 1
OR OR OR OR
B2 B2 B2 B2
F 2 = 2 = 2 = 2
IF FT ? < 2 < 2 > 1 > 9 > 9 < 2 > 1 < 2 > 1 < 2 < 2
AND AND AND AND AND
FT FT FT FT FT
< < < < <
10 20 20 10 10
IF PP > 15 J N J N N J N N N N
COMPUTE X X X X X
DP = GEHALT
/ 100 * 15
COMPUTE X X X X X X X X X
AP = GEHALT
/ 100 *
(20 - 2 * FT)
COMPUTE X X
AP = GEHALT
/ 100 * 5
COMPUTE 20 20 20 20 20 20 10
PP = GEHALT
/ 100 *
PP / 20 * 2
COMPUTE X X X
ZP = GEHALT
/ 100 *
PP / 20 * 2
PERFORM X X X X X X X X X X X X
B-PRAEMIE2
THRU B-ENDE.
ADD DP HP PP
ZP SP GIVING
UEBTRAG.
PERFORM C-DRU.
MOVE 0 TO DP
AP PP ZP SP
GO TO A-LESEN.

Es sei darauf hingewiesen, dass das Tabellenformat vom Programmierer sehr flexibel gehandhabt werden kann. Bedingungen werden durch "C" in Spalte 7, Aktionen durch "A" in Spalte 7 und Regeln durch "R" über der jeweils ersten Spalte einer Regel angezeigt. Sollte die Seite des Codierblattes zur Darstellung der Tabelle nicht ausreichen, können weitere Regeln in Fortsetzungsblöcken untergebracht werden.

Die Codierung einer bereits mit Hilfe der Logikanalyse überprüften Tabelle ist selbst wieder fehleranfällig. Es ist daher zu empfehlen, auch die codierte Tabelle einer Logikanalyse zu unterziehen. Folgender Output wird erstellt:

MBP - V O R E L L E

PROGRAM:JPRAEMIE LOGIKANALYSE

TABLE 002

F T	R 01	R 02	R 03	R 04	R 05	R 06	R 07	R 08	R 09	R 10	R 11	E 12
C 01 IF BZ ?	> 2	> 2	> 2	> 2	> 2	= 1 OR BZ = 2	= 1 OR BZ = 2	= 1 OR BZ = 2	= 1 OR BZ = 2	< 1	< 1	-
C 02 IF FT ?	< 2	< 2	> 1 AND FT < 10	> 9 AND FT < 20	> 9 AND FT < 20	< 2	> 1 AND FT < 10	< 2	> 1 AND FT < 10	< 2	< 2	-
C 03 IF PZ > 15	J	J	-	J	N	J	J	N	N	N	J	-
A 01 COMPUTE DP = GEHALT / 100 * 15	X	X	X	X	X	-	-	-	-	-	-	-
A 02 COMPUTE AP = GEHALT / 100 * (20 - 2 * FT)	X	X	X	-	-	X	X	X	X	-	-	-
A 03 COMPUTE AP = GEHALT / 100 * 5	-	-	-	-	-	-	-	-	-	X	X	-
A 04 COMPUTE PP = GEHALT / 100 * PZ / 20 * 7	20	20	20	20	-	20	20	-	-	-	10	-
A 05 COMPUTE ZP = GEHALT / 100 * PZ / 20 * 2	X	-	-	-	-	X	-	-	-	-	X	-
A 06 PERFORM B-PRAEMIE2 THRU B-ENDE ADD DP AP PP ZP SP GIVING DBETRAG PERFORM C-ORU MOVE 0 TO DP AP PP ZP SP GO TO A-LESEN	X	X	X	X	X	X	X	X	X	X	X	X

```
TABLE    002   DIE ANALYSE BASIERT AUF FOLGENDEN BEDINGUNGSANZEIGERN:

C 01 I > 2 I = 1 I < 1 I
     I     I OR  I     I
     I     I BZ  I     I
     I     I = 2 I     I
-------------------------
C 02 I < 2 I > 1 I > 9 I
     I     I AND I AND I
     I     I FT  I FT  I
     I     I <   I <   I
     I     I 10  I 20  I
-------------------------
C 03 I Y I N I

                        MBP - V O R E L L E          VERSION  COBOL-DOS

PPOGRAM:JPKAEMIE                LOGIKANALYSE

TABLE    002   ERGEBNIS DER VOLLSTAENDIGKEITSPRUEFUNG:
               DIE TABELLE IST FORMAL VOLLSTAENDIG.
               DIE ELSE-REGEL ENTHAELT FOLGENDE KONSTELLATIONEN:

     I 0   I 0   I 0   I 0   I
     I 0   I 0   I 0   I 0   I
     I 1   I 2   I 3   I 4   I
-----+-----+-----+-----+-----+
C 01 I > 2 I = 1 I < 1 I < 1 I
     I     I OR  I     I     I
     I     I BZ  I     I     I
     I     I = 2 I     I     I
-----+-----+-----+-----+-----+
C 02 I < 2 I > 9 I > 1 I > 9 I
     I     I AND I AND I AND I
     I     I FT  I FT  I FT  I
     I     I <   I <   I <   I
     I     I 20  I 10  I 20  I
-----+-----+-----+-----+-----+
C 03 I N   I -   I -   I -   I
-----+-----+-----+-----+-----+


TABLE    002   ERGEBNIS DER REDUNDANZPRUEFUNG:
               DIE TABELLE ENTHAELT REDUNDANZEN UND/ODER WIDERSPRUECHE.

     IR I
     I0 I
     I2 I
-----+--+
R 01 IwII
-----+--+

      R=REDUNDANZ  W=WIDERSPRUCH
      I=IDENTITAET U=UEBERSCHNEIDUNG E=SPALTENREGEL ENTHAELT ZEILENREGEL.


TABLE    002   E N D E  DER ANALYSE
```

Zumindest die in der Tabelle enthaltene Mehrdeutigkeit (Widerspruch) muss entfernt werden. Der Vorübersetzer verfügt über ein Korrektursystem, das zur Fehlerbereinigung eine sehr einfache Korrektursprache anbietet. Auf eine Erläuterung soll hier verzichtet werden.

Nach der Korrektur struktureller Fehler der Entscheidungstabelle kann die eigentliche Vorübersetzung erfolgen. Die Programmliste enthält die ursprüngliche Tabelle als Kommentar, gefolgt von den erzeugten COBOL-Anweisungen.[1)]:

---

1) VORELLE erzeugt auch Vereinbarungen innerhalb der DATA DIVISION, auf deren Erläuterung hier verzichtet wird.

```
00182   001810 B SECTION.                                                                                    000089
00183   001820 B-PRAEMIE1.                                                                                   000090
00184   001830*****************************************************************************T002
00185   001840*     TABLE     002 *HAUPTPRAEMIE                                                              000091
00186   001850*               PRAEMIENBESTANDTEILE:                                                          000092
00187   001860*               DP  DIENSTALTERPRAEMIE        AP  ANWESENHEITSPRAEMIE                          000093
00188   001870*               PP  PRODUKTIVITAETSPRAEMIE    SP  SONDERPRAEMIE                                000094
00189   001880*               ZP  PRAEMIE FUER BESONDERE LEISTUNGEN                                          000095
00190   001890*F   T             R   R   R   R   R   R   R   K   R   R   R   E                              000096
00191   001900*                 0   0   0   0   0   0   0   0   0   1   1   1  T002
00192   001910*                 1   2   3   4   5   6   7   8   9   0   1   2  T002
00193   001920*C01 IF BZ ?       > 2 > 2 > 2 > 2 > 2 = 1 = 1 = 1 = 1 < 1 < 1 -    000097
00194   001930*                                     OR  OR  OR  OR               000098
00195   001940*                                     BZ  BZ  BZ  BZ               000099
00196   001950*                                     = 2 = 2 = 2 = 2              000100
00197   001960*C02 IF FT ?       < 2 < 2 > 1 > 9 > 9 < 2 > 1 < 2 > 1 < 2 < 2 -    000101
00198   001970*                       AND AND AND     AND     AND              000102
00199   001980*                       FT  FT  FT      FT      FT               000103
00200   001990*                       <   <   <       <       <                000104
00201   002000*                       10  20  20      10      10               000105
00202   002010*C03 IF PZ > 15    J   N   -   J   N   J   J   N   N   N   J   -    000106
00203   002020*-------------------------------------------------------------------- T002
00204   002030*A01 COMPUTE       X   X   X   X   X   -   -   -   -   -   -   -    000107
00205   002040*     DP = GEHALT                                                                             000108
00206   002050*     / 100 * 15                                                                              000109
00207   002060*A02 COMPUTE       X   X   X   -   -   X   X   X   X   -   -   -    000110
00208   002070*     AP = GEHALT                                                                             000111
00209   002080*     / 100 *                                                                                 000112
00210   002090*     (20 - 2 * FT)                                                                           000113
00211   002100*A03 COMPUTE       -   -   -   -   -   -   -   -   -   X   X   -    000114
00212   002110*     AP = GEHALT                                                                             000115
00213   002120*     / 100 * 5                                                                               000116
00214   002130*A04 COMPUTE       20  20  20  20  -   20  20  -   -   -   10  -    000117
00215   002140*     PP = GEHALT                                                                             000118
00216   002150*     / 100 *                                                                                 000119
00217   002160*     PZ / 20 * ?                                                                             000120
00218   002170*A05 COMPUTE       X   -   -   -   -   X   -   -   -   -   X   -    000121
00219   002180*     ZP = GEHALT                                                                             000122
00220   002190*     / 100 *                                                                                 000123
00221   002200*     PZ / 20 * 2                                                                             000124
00222   002210*A06 PERFORM       X   X   X   X   X   X   X   X   X   X   X   X    000125
00223   002220*     B-PRAEMIE2                                                                              000126
00224   002230*     THRU B-ENDE                                                                             000127
00225   002240*     ADD DP AP PP                                                                            000128
00226   002250*     ZP SP GIVING                                                                            000129
00227   002260*     OBETRAG                                                                                 000130
00228   002270*     PERFORM C-DRU                                                                           000131
00229   002280*     MOVE 0 TO DP                                                                            000132
00230   002290*     AP PP ZP SP                                                                             000133
00231   002300*     GO TO A-LESEN                                                                           000134
00232   002310*E                                                                                           000135
00233   002320*****************************************************************************T002
00234   002330      MOVE 002-R TO 002-LL. MOVE  0 TO 002-ZNZ.                                  T002
00235   002340      IF BZ > 2                    MOVE -2048 TO 002-L1 (001).   T002C01
00236   002350      IF BZ = 1 OR BZ = 2            ADD 1920 TO 002-L1 (001).   T002C01
00237   002360      IF BZ < 1                        ADD 96 TO 002-L1 (001).   T002C01
00238   002370 002-01-M.                                                       T002C01
00239   002380      IF FT < 2                   MOVE -15008 TO 002-L1 (002).   T002C02
00240   002390      IF FT > 1 AND FT < 10          ADD 6832 TO 002-L1 (002).   T002C02
00241   002400      IF FT > 9 AND FT < 20          ADD 6144 TO 002-L1 (002).   T002C02
00242   002410 002-02-M.                                                       T002C02
00243   002420      IF PZ > 15                  MOVE -18912 TO 002-L1 (003) ELSET002C03
00244   002430                                   MOVE 27072 TO 002-L1 (003).   T002C03
00245   002440 002-VKN.  ENTER LINKAGE.                                                           T002
00246   002450      CALL 'VORELC51' USING 002-TBS 000-TAS 002-TLS. ENTER COBOL.                    T002
00247   002460 002-AA.   MOVE 002-HH TO 002-HHH.                                                  T002
00248   002470      GO TO 002-01-A 002-02-A 002-03-A 002-04-A 002-05-A 002-06-A                    T002
00249   002480           002-07-A 002-AE   DEPENDING ON 002-ZNA.                                   T002
00250   002490 002-01-A. COMPUTE DP = GEHALT / 100 * 15.          GO TO 002-AA.   T002A01
00251   002500 002-02-A. COMPUTE AP = GEHALT / 100 * (20 - 2 * FT).               T002A02
00252   002510                                                    GO TO 002-AA.   T002A02
00253   002520 002-03-A. COMPUTE AP = GEHALT / 100 * 5.           GO TO 002-AA.   T002A03
00254   002530 002-04-A. COMPUTE PP = GEHALT / 100 * PZ / 20 * 20.                 T002A04
00255   002540                                                    GO TO 002-AA.   T002A04
00256   002550 002-05-A. COMPUTE PP = GEHALT / 100 * PZ / 20 * 10.                 T002A04
00257   002560 002-06-A. COMPUTE ZP = GEHALT / 100 * PZ / 20 * 2.                  T002A05
00258   002570 002-07-A. PERFORM B-PRAEMIE2 THRU B-ENDE ADD DP AP PP ZP SP         T002A06
00259   002580      GIVING OBETRAG PERFORM C-DRU MOVE 0 TO DP AP PP ZP SP GO TO    T002A06
00260   002590      A-LESEN.                                                       T002A06
00261   002600 002-AE.   MOVE LOW-VALUE TO 002-TIL.                                            T002
```

Das Programm kann nun mit Hilfe des COBOL-Kompilierers in Maschinensprache übersetzt und im Anschluss daran getestet werden. Im Test leistet eine entscheidungstabellenorientierte Testhilfe wertvolle Dienste, die Entscheidungstabellen-Statistik. Sie zeigt auf, wie oft die Regeln der im Programm enthaltenen Tabellen bei der Bearbeitung der Testdaten durchlaufen wurden, so dass eine Kontrolle der Vollständigkeit der Testdaten möglich ist. Angewendet bei Programmdurchläufen mit echten Daten, übernimmt die Entscheidungsstabellen-Statistik die Funktion eines einfachen Softwaremonitors.

Sie zeigt dann die aktivsten Stellen des Programms auf.

Ein Entscheidungstabellen-Vorübersetzer unterstützt somit alle Phasen der Programmerstellung von der Istaufnahme bis zum Test des Programms. Selbstverständlich leistet er auch bei der Wartung des Programms wertvolle Dienste. Alle während der Programmwartung vorgenommenen Tabellenänderungen werden sofort mit Hilfe der Logikanalyse verifiziert, so dass das Risiko, fehlerhafte Änderungen vorzunehmen, stark begrenzt wird.

5.2 Entscheidungstabellen als Hilfsmittel der organisatorischen Implementierung

Während in der Phase der systemtechnischen Implementierung alle Bestandteile des Informationssystems, die den Rechner als Aktionsträger erfordern, realisiert werden, geschieht das bei der *organisatorischen Implementierung* mit den restlichen Bestandteilen, die den Menschen als Aktionsträger erfordern. Nach einer kurzen Erörterung der in dieser Implementierungsphase anfallenden Teillaufgaben sollen die Einsatzmöglichkeiten der Entscheidungstabellentechnik aufgezeigt werden.

5.2.1 Teilaufgaben der organisatorischen Implementierung

Eine Teilaufgabe der organisatorischen Implementierung ist die Durchführung von Änderungen in der bestehenden Aktionsträgerstruktur. Es kann die Schaffung neuer Stellen erforderlich werden, die in die bestehende Aufbauorganisation eingeordnet werden müssen, oder die Aufgaben bestehender Stellen ändern sich.[1)]

Eine weitere Teilaufgabe ist die Ausarbeitung von Organisationsrichtlinien zur Regelung aller personalen Aktivitäten.[2)]

Weiterhin müssen die personalen Aktionsträger geschult werden, damit sie die ihnen zugewiesenen Funktionen erfüllen können.[3)]

Die Inbetriebnahme des neuen Informationssystems bildet den Abschluss der organisatorischen Implementierung.[4)]

5.2.2 Einsatzmöglichkeiten der Entscheidungstabellentechnik

Entscheidungstabellen lassen sich im Rahmen der organisatorischen Implementierung immer dann einsetzen, wenn komplexe Informationsverarbeitungsprozesse zu erläutern sind. Das kann bei der Aufstellung von Organisationsrichtlinien der Fall sein. Organisationsrichtlinien richten sich an personale Aktionsträger. Sie werden umso wirksamer sein, je präziser und kürzer sie abgefasst sind. Die Entscheidungstabelle als Beschreibungsmittel erleichtert die Erstellung von Organisationsrichtlinien durch ihre wirksamen Möglichkeiten zur Verifikation. Sie stellt eine komplexe Entscheidungslogik zudem in wesentlich kompakterer Form dar, als das eine verbale Beschreibung vermag. [5)] Entscheidungstabellen sind sehr viel einfacher lesbar als verbale Beschreibungen.[6)] Eine relativ kurze Einweisung, die nicht die Tabellenerstellungsmethoden

---

1) Vgl. dazu Schmitz; Seibt (Anwendungsorientierte Informatik), S. 182
2) Vgl. dazu Schmitz; Seibt (Anwendungsorientierte Informatik), S. 183
3) Vgl. dazu Schmitz; Seibt (Anwendungsorientierte Informatik), S. 184 f.
4) Zu den dabei üblicherweise angewendeten Verfahren vgl. Schmitz; Seibt (Anwendungsorientierte Informatik), S. 188 ff.
5) Vgl. dazu Berkey (Clerical procedures), S. 22 ff.
6) Eine Bereitschaft, formale Beschreibungsmethoden zu akzeptieren, muss jedoch vorausgesetzt werden.

und Verifikationsmethoden beinhalten sollte, genügt, um sie als Kommunikationsmittel einsetzen zu können.

Vielfältige Einsatzmöglichkeiten ergeben sich für die Entscheidungsstabelle im Rahmen der Funktionsschulung für die personalen Aktionsträger, die Systembenutzer.

Zum einen müssen dem Benutzer die Methoden der Erfüllung der ihm zugewiesenen Aufgaben erläutert werden. Zwar ist dabei bei weitem nicht der Detaillierungsgrad erforderlich, der bei der Implementierung automatischer Datenverarbeitungsprozesse notwendig ist. Dennoch müssen insbesondere komplexe Entscheidungsvorgänge sehr detailliert erläutert werden, damit dem Benutzer auch alle Sonderfälle transparent werden. Wie kein anderes Hilfsmittel eignet sich hier die Entscheidungstabelle als Kommunikationsmittel zwischen Ausbilder und Auszubildenden.

Zum anderen werden im Rahmen einer Funktionsschulung auch die wichtigsten Teile der automatisch abgewickelten Abläufe erläutert, um dem personalen Aktionsträger den Überblick über die ihn betreffenden Teile des Informationssystems zu geben. Zu diesem Teil der Funktionsschulung können häufig Entscheidungstabellen benutzt werden, die während des systemtechnischen Entwurfs oder der systemtechnischen Implementierung entstanden sind. Ein ausreichender Einblick in die der maschinellen Verarbeitung zugrundeliegenden Algorithmen baut nicht nur psychologische Schranken ab, er gewährleistet auch eine aktive Mitarbeit des Benutzers bei der Lokalisierung noch vorhandener Unzulänglichkeiten des Systems.

6. Auswirkungen der Anwendung von Entscheidungstabellen bei der Systemanwendung

Die Diskussion der Auswirkungen der Anwendung von Entscheidungstabellen bei der Systemanwendung soll sich auf zwei Aspekte beschränken: die Auswirkungen auf die Zuverlässigkeit und die Wartungsfreundlichkeit der Programme des rechnergestützten Informationssystems.

6.1 Zuverlässigkeit der Programme

Die Zuverlässigkeit eines Programms kann definiert werden als die Wahrscheinlichkeit, dass es seine geplante Funktion in einem spezifizierten Zeitraum unter festgelegten Bedingungen erfüllt.[1)] Wenn es möglich wäre, ein Programm vollständig zu verifizieren, dann wäre das Problem der Zuverlässigkeit gelöst. Aus den bereits erläuterten Gründen ist eine vollständige Verifikation jedoch nicht möglich. Häufig wird deshalb die Meinung vertreten, dass zuverlässige Software nicht durch noch intensiveres Testen gewährleistet werden kann, sondern dass bereits in der Entwurfsphase die Weichen für ein zuverlässiges System gestellt werden müssen.[2)] Die Anwendung der Modulartechnik gilt als beste Methode, die Komplexität eines Systems im Interesse einer größeren Zuverlässigkeit zu verringern.[3)]

Ferner gilt die konsequente Anwendung der Strukturierten Programmierung als Mittel der Erzielung einer höheren Programmzuverlässigkeit. Insbesondere wird das darauf zurückgeführt, dass bei ausschließlicher Anwendung der zugelassenen elementaren Ablaufstrukturen bei der Programmerstellung die Ermittlung empirisch zu testender Programmpfade erheblich erleichtert wird.[4)] Das New-York-Times-Projekt der IBM wird in diesem Zusammenhang häufig als Nachweis dieser Eigenschaft der Strukturierten Programmierung zitiert. Ein On-line-Datenbanksystem, das insgesamt 83.324 Ursprungsprogrammzeilen umfasste und nach den Regeln der Strukturierten Programmierung erstellt wurde, wies nach der Übergabe im Zeitraum von etwa eineinhalb Jahren nur 25 Programmfehler auf, was einer Fehlerrate von einem Fehler pro fünf Mannmonaten Pro-

---

1) Vgl. Mac Williams (Introductory) zitiert nach Carey (Software reliability), S. 119
2) Vgl. Liskov (Reliable software systems), S. 192
3) Vgl. Liskov (Reliable software systems), S. 192; Ogdin (Improving software reliability), S. 49 ff.
4) Vgl. Baker (System quality), S. 340

jektarbeit entsprach.[1)]

Die konsequente Anwendung der Entscheidungstabellentechnik scheint ähnlich bemerkenswerte Einflüsse auf die Zuverlässigkeit der Programme auszuüben. Die On-line-Programme zur Automatisierung des Sparverkehrs der Stadtsparkasse Köln, die 1973 mit einem Aufwand von etwa 8 Mannjahren und unter konsequentem Einsatz der Entscheidungstabellentechnik und eines Entscheidungstabellenübersetzers erstellt wurden, erwiesen sich als fast fehlerfrei. Nach Inbetriebnahme des Systems trat ein volles Jahr kein Programmfehler auf, erst danach wurde ein Programmfehler in einer Anweisung *außerhalb* einer Entscheidungstabelle entdeckt. Diese ungewohnte Zuverlässigkeit des Systems wird vorwiegend auf den Einsatz der Entscheidungstabellentechnik zurückgeführt.[2)]

Der positive Einfluss der Anwendung der Entscheidungstabellentechnik auf die Zuverlässigkeit von Programmen lässt sich leicht begründen. Bereits in der Entwurfsphase angewendet, gewährleisten die Verifikationsprozeduren der Entscheidungstabellentechnik in hohem Maße logische Konsistenz und Vollständigkeit der Programmspezifikationen. Während der Systemimplementierungsphase kann der Programmierer auf der Basis eines strukturell einfach zu verifizierenden Programms erheblich wirksamere Tests zur Verifizierung des Programminhalts durchfuhren, als das bei Anwendung konventioneller Programmerstellungsmethoden möglich wäre.

Das Risiko, ganze Programmteile ungetestet zu lassen - eine häufige Ursache späterer Programmzusammenbrüche -, ist bei der rechnergestützten Anwendung der Entscheidungstabellentechnik mit der Aktivierung der Testhilfe "Regelhäufigkeitsstatistik" weitgehend ausgeschaltet. Eine absolute Zuverlässigkeit von Programmen kann die Entscheidungstabellentechnik jedoch nicht garantieren, da sich ihr Einflussbereich auf strukturelle Eigenschaften des Programms beschränkt.

---

1) Vgl. Baker (System quality), S. 342
Andere Faktoren außer der Anwendung der Strukturierten Programmierung haben dieses Ergebnis mit beeinflusst. Insbesondere dürfte sich auch die Organisation der Projektgruppe in der Form des "chief programmer teams" auf das Ergebnis ausgewirkt haben.
Vgl. dazu auch Baker (Chief programmer team), S. 70

2) Vgl. Leckebusch (Entscheidungstabellentechnik)
Das zitierte Interview wurde im April 1974 mit den Verantwortlichen der Stadtsparkasse Köln durchgeführt. Damals war das System erst 3 Monate in Betrieb. In einer Mitteilung an den Verfasser im Januar 1975 wurden die im gesamten Jahr 1974 gemachten Erfahrungen mit oben wiedergegebenem Inhalt bestätigt.

6.2 Wartungsfreundlichkeit der Programme

Jede Programmänderung erfordert eine mehr oder weniger intensive Wiederholung des Systemgestaltungsprozesses, wenn auch in geringerer Dimension.

Die Planungsphase beginnt mit einer Analyse der Ursachen, die eine Programmänderung erforderlich machen. Es lassen sich dabei vier Hauptursachen unterscheiden:[1)]

- Fehler treten auf,
- die Wirksamkeit des Systems muss verbessert werden,
- verbesserte Hardware erfordert eine Anpassung der Software,
- veränderte Benutzeranforderungen erfordern eine Anpassung der Software.

Ein auf die Analysephase folgender Entwurf soll die günstigste Möglichkeit der Durchführung der erforderlichen Programmänderungen bestimmen. Die Implementierungsphase besteht in der Änderung des Programms, der Verifikation, dass die Zielsetzung der Änderung erreicht wurde und dabei keine unerwünschten Nebeneffekte eingetreten sind, und der Inbetriebnahme des geänderten Systems.

Die Programmwartung erfordert einen erheblichen Aufwand. Von durchschnittlich 32 % der Programmiererkapazität [2)] bis etwa 50 % des auf die Programmierung entfallenden Aufwandes [3)] reichen die Angaben zum Aufwand für die Programmwartung, wobei in Einzelfällen auch fast die gesamte Programmiererkapazität zur Programmwartung eingesetzt werden muss.[4)]

Die Anwendung der Entscheidungstabellentechnik kann zwar nicht alle Ursachen für erforderliche Programmänderungen aus der Welt schaffen, sie kann den Aufwand für die laufende Programmwartung jedoch entscheidend senken.

Die zu ändernde Stelle in einem Programm zu lokalisieren, ist bei einem überwiegend aus Entscheidungstabellen bestehenden Programm erheblich einfacher als bei einem konventionell codierten Programm. Das lässt sich

---

1) Vgl. dazu Naur; Randell (Software engineering), S. 111 f.; Canning (Maintenance "iceberg"), S. 6
2) Vgl. Seibt; Stöwer (Erfahrungen der Anwender), S. 247
3) Vgl. Canning (Maintenance "iceberg"), S. 1
4) Vgl. Canning (Maintenance "iceberg"), S. 6

darauf zurückführen, dass es in der Regel genügt, Bedingungen und Aktionen einer Entscheidungstabelle zu lesen, um entscheiden zu können, ob sie von der beabsichtigten Programmänderung betroffen ist oder nicht. Bedingungen und Aktionen sind jedoch nur ein kleiner Teil der insgesamt durch die Entscheidungstabelle abgedeckten Verarbeitungslogik.

Häufig kann die Änderungen fordernde Fachabteilung aber auch bereits die Entscheidungstabelle bezeichnen, die zu ändern ist.[1)]

Falls die Programmänderung durch die Änderung von Entscheidungstabellen realisiert werden konnte, lassen sich unerwünschte Auswirkungen der Änderung durch die sofort anwendbaren strukturellen Verifikationsprozeduren weitgehend verhindern. Nach der Änderung einer Entscheidungsstabelle verschlechtert sich die Programmtransparenz an dieser Stelle nicht. Auch nach häufiger Änderung einer Entscheidungstabelle entsteht kein undurchsichtiges "Flickwerk", ein häufig zu beobachtendes Ergebnis konventioneller Programmerstellungs- und -wartungsmethoden.

Die einfachere Programmwartung bei Einsatz von Entscheidungstabellen ist eine von Anwendern der Entscheidungstabellentechnik häufig bestätigte Erfahrung.[2)] Schätzungen auf der Basis praktischer Erfahrungen besagen, dass die Produktivität von Programmierern, die Wartungsarbeiten durchzuführen haben, mindestens doppelt so hoch ist, verglichen mit der Wartung konventioneller Programme.[3)] Damit kann ein erheblicher Beitrag der Entscheidungstabellentechnik zur Wartungsfreundlichkeit von Programmen als gesichert gelten.

---

1) Vgl. Canning (Maintenance "iceberg"), S. 9
Bei Anwendung der Entscheidungstabellentechnik für technische Problemstellungen (z.B. Konstruktionsprobleme) können Programmänderungen durchaus von Fachabteilungsangehörigen durchgeführt werden. Die Anwendung eines Entscheidungstabellenübersetzers ist dann jedoch Voraussetzung.
Vgl. dazu Brauchle (Entscheidungstabellentechnik), S. 83

2) Bei einer 1970 vom Verfasser durchgeführten Befragung von Entscheidungstabellen-Anwendern in den USA und der Bundesrepublik war die einfachere Programmwartung ein von fast allen Anwendern vorgebrachtes Argument.
Vgl. Strunz (Voruntersuchungen), S. 72 ff.

3) Vgl. Sterbenz (TABSOL), S. 33; Krömer (Programm-Änderungsdienst), S. 96

7. Schluss

Die Entscheidungstabellentechnik findet - so kann abschließend festgestellt werden - Anwendungsmöglichkeiten in fast allen Teilphasen der Gestaltung rechnergestützter Informationssysteme. Verglichen mit anderen in der Praxis eingesetzten Analyse- und Beschreibungshilfsmitteln bietet sie als Strukturspräche ein wesentlich ausgereifteres Bild. Der mittlerweile erreichte Stand in der Entwicklung ihrer theoretischen Grundlagen lässt sie als Werkzeug bei der Systemgestaltung unentbehrlich erscheinen.

Bereits bei *manueller* Anwendung der Entscheidungstabellentechnik ermöglichen die Tabellenerstellungs- und Verifikationsverfahren eine erheblich höhere Genauigkeit bei der Darstellung der Struktur von Entscheidungsprozessen, als das mit anderen Beschreibungstechniken möglich ist. Die daraus zwangsläufig resultierende geringere Fehlerrate in allen Teilphasen der Systemgestaltung bewirkt nicht nur eine höhere Zuverlässigkeit des Informationssystems, sie beeinflusst auch die Kosten der Systementwicklung und -wartung. Die intensive Anwendung der Entscheidungstabellentechnik reduziert logische Fehler in den beschriebenen Informationsverarbeitungsprozessen und vermindert dadurch jene Kategorie von Fehlern, deren Beseitigung sich in der Datenverarbeitungspraxis als außerordentlich kostenintensiv erwiesen hat.

Bei *rechnergestützter* Anwendung der Entscheidungstabellentechnik werden syntaktische Bildungs-, Transformations- und Verifikationsoperationen automatisch durchgeführt. Als zufriedenstellend kann die Rechnerunterstützung in der Entscheidungstabellentechnik jedoch erst bei der Durchführung syntaktischer Operationen an bestehenden Entscheidungsstabellen bezeichnet werden. Der rechnergestützte Aufbau von Entscheidungstabellen bedarf noch intensiver Forschungsarbeiten. Ein möglicher Ansatz für Weiterentwicklungen auf diesem Gebiet wurde mit dem Nachweis der Brauchbarkeit graphentheoretischer Verfahren aufgezeigt.

8. Verzeichnis der Abbildungen

Abb. 1: Entscheidungstabellenverbund ... 25
Abb. 2: Strukturierter Entscheidungsstabellenverbund ... 36
Abb. 3: Ermittlung Fehlender Bedingungsanzeiger -Kombinationen ... 78
Abb. 4: Phasen der Erstellung einer Entscheidungstabelle ... 121
Abb. 5: Formale Phasen des Entscheidungsprozesses ... 199
Abb. 6: Diagrammdarstellung eines Entscheidungstabellenverbundes ... 212
Abb. 7: Ergebnis der Ermittlung starker Komponenten ... 217
Abb. 8: Entscheidungstabellenverbund nach Zergliederung ... 218
Abb. 9: Elementare Ablaufstrukturen ... 235
Abb. 10: Interaction mit DO-UNTIL ... 236
Abb. 11: Fallunterscheidung ... 237
Abb. 12: Auflösung von CASE mit IF-THEN-ELSE ... 242
Abb. 13: Elementare Entscheidungstabellen ... 244
Abb. 14: CASE als Entscheidungstabelle ... 245
Abb. 15: Struktur der Codierung beim Regel – für –Regel -Verfahren ... 256
Abb. 16: Struktur der Codierung beim optimierenden Regel –für-Regel-Verfahren ... 257
Abb. 17: Struktur der Codierung beim Entscheidungsbaumverfahren von Egler ... 259
Abb. 18: Optimierte Entscheidungsbaumstruktur nach Hughes, Shank, Stein ... 260
Abb. 19: Struktur der optimierten Aktionencodierung ... 263
Abb. 20: Reinwald/Soland -Algorithmus: Ergebnis nach Prüfung der ersten Bedingung ... 269
Abb. 21: Reinwald/Soland-Algorithmus: Vollständige Zergliederung ... 270
Abb. 22: Modell der Abstraktionsebenen des Informationssystems ... 275
Abb. 23: Erweitertes Modell der Abstraktionsebenen des Informationssystems ... 281

9. Literaturverzeichnis

ACM (Decision tables symposium)
Proceedings. of the decision tables symposium, Sept. 20-21, New York.
Cosponsered by the Systems Group of the Conference on Data Systems Language (CODASYL) and the Joint Users Group (JUG) of the Association for Computing Machinery. Association for Computing Machinery, New York 1962

Air Force (Pamphlet 5-5)
Department of the Air Force:
Headquarters US Air Force: AF Pamphlet 5-5.
Washington DC 15. März 1974

Armerding (FORTAB)
Armerding, G. W.:
FORTAB: A decision table language for scientific computing applications.
In: Proceedings of the Decision Tables Symposium, Sept. 20-21, New York 1962,·S. 81 - 87

Armstrong (Modular programming)
Armstrong, Russell M.:
Modular programming in COBOL.
New York-London-Sydney-Toronto 1973

Asser (Mathematische Logik)
Asser, G.:
Einführung in die Mathematische Logik. Teil I: Aussagenkalkül.
4. unveränderte Aufl. Leipzig 1972

Baker (System quality)
Baker, F. T.:
System quality through structured programming.
In: Proceedings of the Fall Joint Computer Conference 1972, S. 339 - 343

Baker (Chief programmer team)
Baker, F. T.:
Chief programmer team management of production programming.
In: IBM Systems Journal, Bd. 11, 1972, Nr. I, S. 56 - 73

Barnard (NITA)
Barnard, T. J.:
NITA user's guide.
The National Computing Centre Limited,
Manchester 1967

Barnard (New rule mask technique)
Barnard, T. J.:
A new rule mask technique for interpreting decision tables.
In: Computer Bulletin, Bd. 13, 1969, Nr. 5, S. 153 - 154

Baun (Entscheidungstabellen)
Baun, Wulf:
Entscheidungstabellen - Eine Methode zur Analyse und Programmierung von systematischen Abläufen. (IBM-Fachbibliothek). IBM-Form 81570.
Sindelfingen 1969

Beizer (Architecture)
Beizer, Boris :
The architecture and engineering of digital computer complexes. Vol. 1.
New York-London 1971

Bell of Canada (Decision tables)
Bell of Canada:
Decision tables.
Unveröffentlichte Lehrgangsunterlagen, o. 0., o. J.
(ca. 1970)

Berkey (Clerical procedures)
Berkey, Marsha A.:
Decision tables for clerical procedures.
In: Management Accounting, Okt. 1973, S. 21 - 26

Bingham; Davies (Systems analysis)
Bingham, J. E.; Davies, G. W. P.:
A handbook of systems analysis.
London and Basingstoke 1972

Björk (Decision tables in ALGOL 60)
Björk, Harry:
Decision tables in ALGOL 60.
In: BIT, Bd. 8, 1968, Nr. 3, S. 147 - 1~3

Bochenski (Denkmethoden)
Bochenski, I. M.:
Die zeitgenössischen Denkmethoden.
3. Aufl., Bern und München 1965

Boehm (Software and its impact)
Boehm, Barry W.:
Software and its impact: A quantitative assessment.
In: Datamation, Bd. 19, 1973, Nr. 5, S. 48 - 59

Böhm; Jacopini (Flow diagrams)
Böhm, Corrado; Jacopini, Giuseppe:
Flow diagrams, Turing machines and languages with only
two formation rules.
In: Communications of the ACM, Bd. 9, 1966, Nr. 5,
S. 366 - 371

Brauchle (Entscheidungstabellentechnik)
Brauchle, Herbert:
Praxis der Entscheidungstabellentechnik.
Baden-Baden und Bad Homburg vor der Höhe 1973

Brown (Decision table experience) .
Brown, Lynn:
Decision table experience on a file maintenance system.
In: Proceedings of the Decision Tables Symposium,
Sept. 20-21, New York 1962, S. 75 - 80

Caldwell (Entwurf von Schaltkreisen)
Caldwell, Samuel H.:
Der logische Entwurf von Schaltkreisen.
München-Wien 1964

Calkins (DETAB-X)
Calkins, L. W.:
Place of decision tables and DETAB-X.
In: Proceedings of the Decision Tables Symposium,
Sept. 20-21, New York 1962, S. 9 - 12

Calkins (Question and answer)
Calkins, L. W. (Moderator):
Question and answer period. Decision Tables Symposium,
Morning of Sept. 20, 1962. Panel: Burton Grad, Mary K.
Hawes, Solomon L. Pollack, Kendall Wright.
In: Proceedings of the Decision Tables Symposium,
Sept. 20-21, New York 1962, S. 45 - 54

Callahan; Chapman (DETAB/65 preprocessor)
Callahan, Michael D; Chapman, Anson E.:
Description of basis algorithm in DETAB/65 preprocessor.
In: Communications of the ACM, Bd. 10, 1967, Nr. 7,
S. 441 - 446

Canning (Modular COBOL programming)
Canning, Richard (Hrsg.):
Modular COBOL programming.
In: EDP Analyzer, Bd. 10, 1972, Nr. 7

Canning (Maintenance "iceberg")
Canning, Richard (Hrsg.):
That maintenance "iceberg".
In: EDP Analyzer, Bd. 10, 1972, Nr. 10

Canning (Advent of structured programming)
Canning, Richard (Hrsg.):
The advent of structured programming.
In: EDP Analyzer, Bd. 12, 1974, Nr. 6

Cantrell (Applications)
Cantrell, H. N.:
Commercial and engineering applications of decision.
tables.
In: Proceedings of the Decision Tables Symposium,
Sept. 20-21, New York 1962, S. 55 - 60

Carey (Software reliability)
Carey, Levi J.:
IEEE Symposium on software reliability.
In: Datamation, Bd. 19, 1973, Nr. 10, S. 119 und 125

Cavouras (Conversion)
Cavouras, John C.:
On the conversion of programs to decision tables.
Method and objectives.
In: Communications of the ACM, Bd. 17, 1974, Nr. 8,
S. 456 - 462

Chandor; Graham; Williamson (Systems analysis)
Chandor, Anthony; Graham, John; Williamson, Robin:
Practical systems analysis.
London 1969

Chapin (Parsing)
Chapin, Ned:
Parsing of decision tables.
In: Communications of the ACM, Bd. 10, 1967, Nr. 8, S. 507 - 512

Clark (GOTO-less programming)
Clark, R. Lawrence:
A linguistic contribution to GOTO-less programming.
In: Datamation, Bd. 19, 1973, Nr. 12, S. 62 - 63

Clifton (Systems analysis) Clifton , H. D.:
Systems analysis for business data processing.
Princeton-New York-Philadelphia-London 1970

Cohen (Defining the module) Cohen, Alan:
Modular Programs: Defining the module.
In: Datamation, Bd. 18, 1972, Nr. 1, S. 34 - 37

Dahl; Hoare (Hierarchical program structures) Dahl, Ole-Johan; Hoare, C. A. R.:
Hierarchical program structures.
In: Dahl, O.-J; Dijkstra, E. W.; Hoare, C. A. R.:
Structured Programming.
London-New York 1972, S. 175 - 220

Dathe (Conversion of decision tables)
Dathe, Gert:
Conversion of decision tables by rule mask method
without rule mask.
In: Communications of the ACM, Bd. 15, 1972, Nr. 10,
S. 906 - 909

Dent (Decision tables)
Dent, John L.:
Decision tables. A state-of-the-art report.
Monograph # 2, Working paper series der School of
Business Administration,
University of Minnesota. O. O., o. J. (um 1969)

Devine (Decision tables)
Devine, Donald:
Decision tables as the basis of a programming language.
In: Data Processing, Vol.VIII. Proceedings of the 1965
International Data Processing Conference. Data
Processing Management Association.
Philadelphia, Pa. 1965, S. 461 - 466

Dickman (GOTO controversy)
Dickman, B. N.:
More fuel for the GOTO controversy.
In: ACM SIGPLAN Notices, Juni 1974, S. 17

Dijkstra (GO TO statement)
Dijkstra, Edsger W.:
GO TO statement considered harmful.
In: Communications of the ACM, Letter to the Editor, .
Bd. 11, 1968, Nr. 3, S. 147 - 148

Dijkstra ("THE"-multiprogramming system)
Dijkstra, Edsger W.:
The structure of the "THE"-multiprogramming system.
In: Communications of theACM, Bd. 11, 1968, Nr. 5,
S. 341 - 346

Dijkstra (Structured programming)
Dijkstra, E. W.:
Structured programming.
In: Software Engineering Techniques, hrsg. von J. N.
Buxton und B. Randell.
NATO Scientific Affairs Division, Brüssel 1970,
S. 84 - 88

Dijkstra (Humble programmer)
Dijkstra, Edsger W.:
The humble programmer.
In: Communications of the ACM, Bd. 15, 1972, Nr. 10,
S. 859 - 866

Dijkstra (Notes on structured programming)
Dijkstra, Edsger W.:
Notes on structured programming.
In: Dahl, O.-J.; Dijkstra, E. W.; Hoare, C. A. R.:
Structured programming. London-New York 1972,
S. 1 - 82

DNA (Norm-Entwurf DIN 66 220)
Fachnormenausschuss Informationsverarbeitung (FNI) im
Deutschen Normenausschuss (DNA):
Programmablauf für die Verarbeitung von Dateien nach
Satzgruppen. DIN 66 220. Entwurf. Nov. 1974

DNA (Normen)
Deutscher Normenausschuss e.V. (DNA) (Hrsg.):
Normen über Informationsverarbeitung.
3. Aufl., Berlin-Köln-Frankfurt (Main) 1975

Donnell (Decision table checks)
Donnell, Ron:
Decision table checks.
CODASYL-DTTG paper. Unveröffentlichtes Manuskript,
23. Mai 1974

Egler (Converting logic table conditions)
Egler, J. F.:
A procedure for converting logic table conditions into
an efficient sequence of test instructions.
In: Communications of the ACM, Bd. 6, 1963, Nr. 8,
S. 510 - 514

Evans (Advanced analysis method)
Evans, Orren Y.:
Advanced analysis method for integrated electronic data
processing.
IBM General Information Manual, F 20-8047

Fergus (Decision tables)
Fergus, Raymond M.:
Decision tables - An application analyst/programmer view.
In: Data Processing. Vol. XII. Proceedings of the 1967
International Data Processing Conference and Business
Exposition. Data Processing Management Association.
June 20-23, 1967, Boston, Mass. S. 85 - 109

Floyd (Strukturierte Programmierung)
Floyd, Christiane:
Strukturierte Programmierung für COBOL Anwender.
Hamburg 1974

Floyd (Assigning meanings to programs)
Floyd, R. W.:
Assigning meanings to programs.
In: Proceedings of a Symposium in Applied Mathematics,
Vol. 19, hrsg. von J. T. Schwartz, American Mathematical
Society, 1967, S. 19 - 32

FNI (Entscheidungstabelle)
Normenausschuss Informationsverarbeitung (FNI):
Entscheidungstabelle für den Informationsaustausch.
Beschreibungsmittel. Norm-Entwurf DIN 66241

Frese (Kontrolle)
Frese, Erich:
Kontrolle und Unternehmungsführung. Entscheidungs- und
organisationstheoretische Grundfragen.
Wiesbaden 1968

Gellert; Küstner; u. a. (Mathematik)
Gellert, W.; Küstner, H.; Hellwich,M.; Kastner, H.
(Hrsg.) :
Mathematik. Kleine Enzyklopädie.
Frankfurt-Zürich 1972

General Electric (GECOM II)
General Electric:
GECOM-II, Reference manual. Kap. 11, TABSOL, S. 175 - 192

Gildersleeve (Decision tables)
Gildersleeve, Thomas R.:
Decision tables and their practical application in data
processing.
Englewood Ciffs, N. J. 1970

Good; London; Bledsoe (Interactive program verification)
Good, Donald I.; London, Ralph L.; Bledsoe, W. W.:
An interactive program verification system.
Manuskript. Beitrag zur International Conference on
Reliable Software, April 21-23, 1975, Los Angeles

Goodenough; Ross; Eanes (MAIDS)
Goodenough, John B.; Ross, D. T.; Eanes, R. Sterling:
MAIDS. Information Dynamics Technology Requirements
Study. Final Report April 1973. Prepared under contract
DAAA 25-72 C 0667 by Softech Inc, Waltham, Mass. and
submitted to Frankford Arsenal

Grad (Tabular form)
Grad, Burton:
Tabular form in decision logic.
In: Datamation, Bd. 7,1961, Nr. 7, S. 22 - 26

Grochla; Garbe; u. a. (Kölner Integrationsmodell)
Grochla, Erwin; Garbe, Helmut; Gillner, Reinhard;
Poths, Willi:
Das Kölner Integrationsmodell.
In: Erwin Grochla und Mitarbeiter: Integrierte
Gesamtmodelle der Datenverarbeitung. Entwicklung und
Anwendung des Kölner Integrationsmodells (KIM).
München-Wien 1974, S. 189 - 422

Grochla (Konzept des Kölner Integrationsmodells)
Grochla, Erwin:
Das Konzept des Kölner Integrationsmodells.
In: Erwin Grochla und Mitarbeiter: Integrierte
Gesamtmodelle der Datenverarbeitung. Entwicklung und
Anwendung des Kölner Integrationsmodells (KIM).
München-Wien 1974, S. 34 - 46

Harary; Norman; Cartwright (Structural models)
Harary, Frank; Norman, Robert, Z.; Cartwright, Dorwin:
Structural models: an introduction to the theory of
directed graphs.
New York-London-Sydney 1965

Harary (Graphentheorie)
Harary, Frank:
Graphentheorie.
München-Wien 1974

Harris (Decision table macros)
Harris, Edward R.:
ALP DOS Decision table macros.
International Business Machines Corporation. Contributed program library, Nr. 360D-03.7.017, 1967

Harrison (Complete decision tables)
Harrison, William J.:
Practically complete decision tables: A range approach.
In: ACM SIGPLAN Notices, Bd. 6, 1971, Nr. 8, S. 89 - 93

Hartman; Matthes; Proeme (Systems handbook)
Hartman, W.; Matthes, H.; Proeme, A.:
Information systems handbook. Analysis, requirements determination, design and development, implementation and evaluation. 2. Aufl.
Apeldoorn, The Netherlands 1972

Hetzel (Program testing)
Hetzel, William C.:
Principles of computer program testing.
In: Program test methods, hrsg. von William C. Hetzel.
Englewood Cliffs, N. J. 1973, S. 17 - 28

Hoare (Data structuring) Hoare, ·C. A. R.:
Notes on data structuring.
In: Dahl, O.-J.; Dijkstra, E.W.; Hoare, C. A. R.:
Structured programming. London-New York 1972,
S. 83 - 174

Hoernes; Heilweil (Logik-Entwurf)
Hoernes, Gerhard E.; Heilweil, Melvin F.:
Boolsche Algebra und Logik-Entwurf.
München-Wien 1968

Hopkins (Case for the GOTO)
Hopkins, Martin E.:
A case for the GOTO.
In: Proceedings of the ACM Annual Conference.
Boston, Mass. 1972, S. 787 - 790

Hughes; Shank, Stein (Decision tables)
Hughes, Marion L.; Shank, Richard M.; Stein, Elinor Svendsen:
Decision tables.
New York-St. Louis-San Francisco-London-Sydney-Toronto 1968

IBM (Problemlösung)
International Business Machines Corporation:
Methoden zur Problemlösung.
Teilnehmerunterlagen zu einem 2-tagigen Lehrgang.
Nummer der Lehrgangsart: 6304, o. 0., o. J.

IBM (Decision tables)
International Business Machines Corporation:
Decision tables. A systems analysis and documentation technique.
IBM Form F20-8102-0. White Plains, N. Y. 1962

Igarashi; London; Luckham (Automatic program verification)
Igarashi, Shigeru; London, Ralph L.; Luckham, David C.:
Automatic program verification I: A logical basis and its implementation.
Stanford Artifical Intelligence Laboratory Memo AIM-200, STAN-CS-73-365.
Stanford University 1973

Kavanagh (TABSOL)
Kavanagh, T. F.:
TABSOL. A fundamental concept for systems-oriented languages.
In: Proceedings of the Eastern Joint Computer Conference.
New York 1960, S. 117 - 136

Kavanagh (Manufacturing applications)
Kavanagh, T. F.:
Manufacturing applications of decision structure tables.
In: Proceedings of the Decision Tables Symposium, Sept. 20-21, New York 1962, S. 89 - 97

King (Conversion of decision tables)
King, P. J. H.:
Conversion of decision tables to computer programs by rule mask techniques.
In: Communications of the ACM, Bd. 9, 1966, Nr. 11, S. 796 - 801

King (Ambiguity)
King, P. J. H.:
Ambiguity in limited entry decision tables.
In: Communications of the ACM, Bd. 11, 1968, Nr. 10, S. 680 - 684

King; Johnson (Ambiguous decision tables)
King, P. J. H.; Johnson R. G.:
Some comments on the use of ambiguous decision tables and their conversion to computer programs.
In: Communications of the ACM, Bd. 16, 1973, Nr. 5, S. 287 - 290

Kirk (Use of decision tables)
Kirk, H. W.:
Use of decision tables in computer programming.
In: Communications of the ACM, Bd. 8, 1965, Nr. 1, S. 41 - 43

Knuth (Computer programming) Knuth, Donald E.:
The art of computer programming.
Vol. 1: Fundamental algorithms, Reading, Mass. 2. Aufl. 1973
Vol. 2: Seminumerical algorithms, Reading, Mass. 2. Aufl. 1971
Vol. 3: Sorting and searching, Reading, Mass. 1973

Knuth (Structured programming)
Knuth, Donald E.:
Structured programming with GO TO statements.
In: ACM Computing Surveys. Bd. 6, 1974, Nr. 4, S.
261 - 301

Koster; Hetzel (Datenverarbeitung mit System)
Koster, Walter; Hetzel, Friedrich:
Datenverarbeitung mit System.
Neuwied-Berlin 1971

Kosiol (Organisation)
Kosiol, Erich:
Organisation der Unternehmung.
Wiesbaden 1962

Krömer (Programm-Änderungsdienst)
Krömer, Herbert:
Rationalisieren Entscheidungstabellen den
Programmm-Änderungsdienst? Ein Beispiel aus der Praxis.
In: Online, 11. Jg. 1973, Nr. 11, S. 790 - 796

Lauer (Auflösung von Entscheidungstabellen)
Lauer, H.:
Formalismen zur Auflösung von Entscheidungstabellen.
In: Angewandte Informatik, 16. Jg. der "elektronischen
datenverarbeitung" 1974, Heft 10, S. 437 - 443

Langefors (Information systems)
Langefors, Borje:
Theoretical analysis of information systems.
4. Aufl. Philadelphia, Pa. 1973

Leavenworth (Programming with(out) the GO TO)
Leavenworth, B. M.:
Programming with(out) the GO TO.
In: Proceedings of the ACM Annual Conference,
Boston, Mass. 1972, S. 782 - 786

Leckebusch (Entscheidungstabellentechnik)
Leckebusch, Norbert:
Entscheidungstabellentechnik bei der Stadtsparkasse Köln.
Sonderdruck aus: Die Computer-Zeitung, Ausgabe 10 vom
15. Mai 1974

Lee (Modulare Programmierung)
Lee, J.:
Kosten senken mit modularer Programmierung.
In: Online, 12. Jg. 1974, Nr. 9, S. 531 - 536

Liskov (Reliable software systems)
Liskov, B. H.:
A design methodology for reliable software systems.
In: Proceedings of the Fall Joint Computer Conference
1972, S. 191 - 199

London (Decision tables)
London, Keith, R.:
Decision tables.
Princeton-Philadelphia-New York-London 1972

London (Proving programs correct)
London, Ralph L.:
The current state of proving programs correct.
In: Proceedings of the ACM Annual Conference 1972, S. 39 - 46

London (Program verification) London, Ralph L.:
A view of program verification.
Manuskript. Beitrag zur International Conference on
Reliable Software, April 21-23, 1975, Los Angeles

Loveman (CTl)
Loveman, David B.:
CTl-A programming language for automatic testing.
Forschungsbericht der Massachusetts Computer Associates,
Inc., Wakefield, Mass. 14. Sept. 1973

Mac Williams (Introductory)
Mac Williams, W. H.:
Introductory.
In: IEEE Symposium on software reliability, New York City,
Mai 1973, zitiert nach Carey (Software reliability)

Mangoldt; Knopp (Höhere Mathematik)
Mangoldt, H. V.; Knopp, Konrad:
Einführung in die Höhere Mathematik.
Erster Band, 12. Aufl. Stuttgart 1964

Maynard (Modular programming)
Maynard, Jeff:
Modular programming.
London 1972

McDaniel (Software)
McDaniel, Herman:
Decision table software.
Princeton-New York-London 1970

mbp (VORELLE-Handbuch)
Mathematischer Beratungs- und Programmierungsdienst GmbH:
Benutzerhandbuch. VORELLE. Vorübersetzer für
Entscheidungstabellen. COBOL-Version.
Dortmund 1974

Miller; Lindamood (Top-down approach)
Miller, Edward F. Jr.; Lindamood, George E.:
Structured programming: Top-down approach.
In: Datamation, Bd. 19, 1973, Nr. 12, S. 55 - 57

Mills (Structured programming)
Mills, H. D.:
Structured programming.
In: Structured programming handouts.
Manuskriptsammlung der International Business Machines
Corp., 0.0., o. J., S. 0 - 69

Mills (Top down programming)
Mills, Harlan:
Top down programming in large systems.
In: Debugging techniques in large systems, hrsg. von
Randall Rustin.
Englewood Cliffs, N. J. 1971, S. 41 - 55

Mills (Chief programmer teams)
Mills, H. D.:
Chief programmer teams. Principles and procedures.
International Business Machines Corporation.
Federal Systems Division, Report FSC-71-5108
Gaithersburg, Maryland 1971

Mills (New math)
Mills, Harlan D.:
The new math of computer programming.
In: Communications of the ACM, Bd. 18, 1975, Nr. 1,
S. 43- 48

Montalbano (Tables, flow charts)
Montalbano, M.:
Tables, flow charts, and programm logic.
In: IBM Systems Journal, Sept. 1962, S. 51 - 63

Montalbano (Egler's procedure)
Montalbano, Michael:
Egler's procedure refuted.
Letter to the editor.
In: Communications of the ACM, Bd. 7, 1964, Nr. 1, S. 1

Montalbano (Decision tables)
Montalbano,. Michael:
Decision tables.
Chicago-Palo Alto-Toronto-Henley-on-Thames-Sydney-
Paris 1974

Morris (Theory of signs)
Morris, Charles:
Foundations of the theory of signs.
In: Foundations of the Unity of Science.
Bd. 1, Nr. 2, Chicago 1938, 12. Aufl. 1970

Muntz (Multiple choice decision tables)
Muntz, Charles:
Multiple choice decision tables.
Forschungsbericht der Massachusetts Computer Associates,
Inc., Wakefield, Mass. 21. Dez. 1973

Muthukrishnan; Rajaraman (Conversion of decision tables)
Muthukrishnan, C. R.; Rajaraman, V.:
On the conversion of decision tables to computer programs.
In: Communications of the ACM, Bd. 13, 1970, Nr. 6, S.
347 - 351

Myers (Compiling optimized code)
Myers,H.J.:
Compiling optimized code from decision tables.
In: IBM Journal of Research and Development,
Sept. 1972, S. 489 - 503

Myers (Composite design)
Myers, Glenford J.:
Composite design: The design of modular programs.
IBM Technical Report TR 00.2406
Poughkeepsie 1973

Naramore (Application)
Naramore, Frederick:
Application of decision tables to management information
systems.
In: Proceedings of the Decision Tables Symposium,
Sept. 20-21, New York 1962, S. 63 - 74

Naur; Randell (Software engineering)
Naur, Peter; Randell, Brian (Hrsg.):
Software engineering. Report on a conference sponsored
by the NATO Science Committee. O. O. 1969

NCC (Decision tables)
The National Computing Centre Limited:
Decision tables in data processing. A report.
Manchester 1970

Nickerson (Engineering application)
Nickerson, R. C.:
An engineering application of logic-structure tables.
In: Communications of the ACM, Bd. 4, 1961, Nr. 11,
S. 516 - 520

O'Brien (Computerizing administrative operations)
O'Brien, James L.:
Some promising approaches to computerizing administrative
operations.
In: Seminar on decision tables, Sept. 30, 1964.
U.S. Department of Commerce, Bureau of the Census.
o. 0., o. J., S. 65 - 86

Ogdin (Improving software reliability)
Ogdin, Jerry L.:
Improving software reliability.
In: Datamation, Bd. 19, 1973, Nr. 1, S. 49 - 52

o. V. (COBOL support packages) o. V.
COBOL support packages. Decision table translators ...
Generating COBOL logic statements.
In: Data Processing Digest, Bd. 17, 1971, Nr. 11

o. V. (Using modular programming) o. V.
Evaluation of programming and systems techniques.
Implications of using modular programming.
Central Computer Agency Guide No.1,
Central Computer Agency, Civil Service Department.
London 1973

Pärli (Istaufnahme)
Studienkreis Dr. Pärli:
Istaufnahme und automatisierte Datenverarbeitung.
Die Istaufnahme eines Datenverarbeitungsproblems
als Teil der organisatorischen Vorarbeiten im
Hinblick auf die Verwendung automatisierter
Datenverarbeitungsverfahren. Wiesbaden 1972

Pooch (Translation of decision tables)
Pooch, Udo W.:
Translation of decision tables.
In: Computing Surveys, Bd. 6, 1974, Nr. 2,
S. 125 - 151

Pollack (DETAB-X)
Pollack, Solomon L.:
What is DETAB-X?
In: Proceedings of the Decision Tables Symposium,
Sept. 20-21, New York 1962, S. 29 - 39

Pollack (CODASYL)
Pollack, Solomon L.:
CODASYL, COBOL & DETAB-X.
In: Datamation, Bd. 9, 1963, Nr. 2, S. 60 - 61

Pollack (Analysis)
Pollack, Solomon L.:
Analysis of the decision rules in decision tables.
Rand-Memorandum RM-3669-PR, May 1963

Pollack (Conversion of limited-entry decision tables)
Pollack, Solomon L.:
Conversion of limited-entry decision tables
to computer programs.
In: Communications of the ACM, Bd. 8, 1965,
Nr. 11, S. 677 - 682

Pollack (Stellungnahme zu Spragues Einwand)
Pollack, Solomon L.:
Ohne Titel.
Stellungnahme zu Spragues Einwand.
In: Communications of the ACM, Letter to the editor,
Bd. 9, 1966, Nr. 5, S. 320

Pollack (Decision tables)
Pollack, Solomon L.:
Decision tables directly into program.
In: Ideas for Management.
Systems and Procedures Associations, o. 0., 1966,
S. 54 - 89

Pollack; Wright (Data decription)
Pollack, Solomon L.; Wright, Kendall R.:
Data description for DETAB-X (Decision Table, Experimental).
RAND Memorandum RM-3010-PR, März 1962

Pollack; Hicks; Harrison (Decision tables)
Pollack, Solomon L.; Hicks, Harry T. Jr.; Harrison, William J.:
Decision tables: Theory and practice.
New York-London-Sydney-Toronto 1971

Press (Conversion of decision tables)
Press, Laurence I.:
Conversion of decision tables to computer programs.
In: Communications of the ACM, Bd. 8, 1965, Nr. 6, S. 385 - 390

Rabin (Optimal decision trees)
Rabin, Jonas:
Conversion of limited-entry decision tables into
optimal decision trees.
In: ACM SIGPLAN Notices, Bd. 6, 1971, Nr. 8, S. 68 - 74

Rabin (DTTG progress report)
Rabin, Jonas:
Decision Table Task Group Progress Report.
Unveröffentlicht, 29. April 1974

Reinwald (Parsing decision tables into computer programs)
Reinwald, Lewis T.:
On parsing decision tables into computer programs
having minimal testing costs.
Unveröffentlichtes Manuskript. o.O., o. J. (nach 1966)

Reinwald (Re-conversion problem)
Reinwald, Lewis T.:
The re-conversion problem (i.e the mapping of a "conventional" program into a set of decision tables).
CODASYL DTTG paper. Unveröffentlichtes Manuskript, 11. März 1974

Reinwald (Progress on automatic conversion)
Reinwald, Lewis T.:
Progress on the automatic conversion of procedural languages programs into decision tables.
CODASYL DTTG paper. Unveröffentlichtes Manuskript, 20. Mai 1974

Reinwald; Soland (Conversion of limited-entry decision tables I)
Reinwald, Lewis T.; Soland, Richard M.:
Conversion of limited-entry decision tables to optimal computer programs I: Minimum average processing time. In: Journal of the Association for Computing Machinery, Bd. 13, 1966, Nr. 3, S. 339 - 358

Reinwald; Soland (Conversion of limited-entry decision tables II)
Reinwald, Lewis T.; Soland, Richard M.:
Conversion of limited-entry decision tables to optimal computer programs II: Minimum storage requirement.
In: Journal of the Association for Computing Machinery, Bd. 14, 1967, Nr. 4, S. 742 - 755

Schmitz; Seibt; u. a. (Modell der Wirksamkeit)
Schmitz, Paul; Seibt, Dietrich; Matis, Heinz; Strunz, Horst:
Entwicklung eines Modells zur Bestimmung der Wirksamkeit von Programmiersprachen.
In: Studienkreis Paul Schmitz: Die Wirksamkeit von Programmiersprachen.
Wiesbaden 1972, S. 14 - 153

Schmitz; Seibt (Anwendungsorientierte Informatik)
Schmitz, Paul; Seibt, Dietrich:
Einführung in die anwendungsorientierte Informatik.
München 1975

Seibt; Stöwer (Erfahrungen der Anwender)
Seibt, Dietrich; Stöwer, Emil:
Erfahrungen der Anwender von ADV-Systemen. Ergebnisse einer empirischen Erhebung.
In: Studienkreis Paul Schmitz: Die Wirksamkeit von Programmiersprachen.
Wiesbaden 1972, S. 218 - 261

Shwayder (Modification to Pollack's algorithm)
Shwayder, Keith:
Conversion of limited-entry decision tables to computer programs - A proposed modification to Pollack's algorithm.
In: Communication of the ACM, Bd. 14, 1971, Nr. 2, S. 69 - 73

SIGPLAN (DETAB/65)
Special Interest Group on Programming Languages (SIGPLAN) of the Los Angeles Chapter of the Association of Computing Machinery, Working group 2: The DETAB/65 language.
O. 0., o. J. (1965)

Simon (Management decision)
Simon, Herbert A.:
The new science of management decision. New York 1960

software ag (agenta) software ag:
agenta, aiv-Generator auf der Grundlage von Entscheidungstabellen – Einführung -. Ergänzende Unterlage zum Vortrag G. Dathe: Mehrdeutige Entscheidungstabellen. Jahreskolloquium für Rechentechnik in Braunschweig am 29. Februar 1972

Sprague (Storage space)
Sprague, V. G.:
On storage space of decision tables.
In: Communications of the ACM, Letters to the Editor.
Bd. 9, 1966, Nr. 5, S. 319

Standards Committee (Revised X3.23 COBOL)
American National Standards Committee X3-Computers and
Information Processing:
Draft Proposed Revised X3.23 American National Standard
Specifications for COBOL.
Washington D.C. August 1972

Sterbakov (Last 'Y' method)
Sterbakov, C. R.:
Last 'Y' method of designing decision tables.
CODASYL-DTTG paper. Unveröffentlichtes Manuskript
20. 5. 1974

Sterbenz (TABSOL)
Sterbenz, Robert F.:
TABSOL decision table preprocessor.
In: ACM SIGPLAN Notices, Bd. 6, 1971, Nr. 8, S. 33 - 40

Strunz (Voruntersuchungen)
Strunz, Horst:
Voruntersuchungen zur Entwicklung einer
Entscheidungstabellensprache und eines
Entscheidungstabellenvorübersetzers.
Unveröffentlichter Forschungsbericht, eingereicht beim
Bundesministerium für Bildung und Wissenschaft.
Dortmund, Dez. 1970

Strunz (Zergliederung)
Strunz, Horst:
Eine Methode zur Zergliederung von Entscheidungstabellen.
BIFOA-Arbeitsbericht 70/12, Köln 1970

Strunz (Entscheidungstabellen)
Strunz, Horst:
Entscheidungstabellen und ihre Anwendung bei Systemplanung,
-implementierung und -dokumentation.
In: elektronische datenverarbeitung, 12. Jg. 1970,
Heft 2, S. 56 - 65

Strunz (Eine Methode zur Zergliederung)
Strunz, Horst:
Eine Methode zur Zergliederung von Entscheidungstabellen.
In: Angewandte Informatik, 13. Jg. der "elektronischen datenverarbeitung" 1971, Heft 3, S. 117 - 122

Strunz (Handbuch 1972)
Strunz, Horst:
Handbuch der Entscheidungstabellentechnik.
Teilnehmer-Unterlagen zu Entscheidungstabellen-Seminaren.
o.O., o. J. (Version 1972)

Strunz (COBOL)
Strunz, Horst:
COBOL. Untersuchungsergebnisse zur Entwicklung, Standardisierung und Implementierung.
In: Studienkreis Paul Schmitz: Die Wirksamkeit von Programmiersprachen.
Wiesbaden 1972, S. 309 - 335

Strunz (Parsing)
Strunz, Horst:
The development of decision tables via parsing of complex decision situations.
In: Communications of the ACM, Bd. 16, 1973, Nr. 6, S. 366 - 369

Strunz (Minimization)
Strunz, Horst:
Minimization of multiple hit tables.
CODASYL-DTTG working paper.
Unveröffentlichtes Manuskript, 1. Juli 1974

Strunz (Decision table networks)
Strunz, Horst:
Decision table networks.
CODASYL DTTG paper. Unveröffentlichtes Manuskript, 8. Nov. 1974

Strunz (Verträglichkeit der Strukturierten Programmierung)
Strunz, Horst:
Verträglichkeit der Strukturierten Programmierung mit anderen Techniken der methodischen Programmentwicklung.
Vortrag auf STRUCTO 76, Kongress für Strukturierte Programmierung vom 26. - 28. Okt. 1976 in Frankfurt

Strunz; Bischoff; u. a. (Gestaltung von Informationssystemen)
Strunz, Horst; Bischoff, Rainer; Nick, Frank; Schaedel, Volker; Schaefer, Richard; Tollert, Arthur:
Möglichkeiten und Grenzen der computergestützten Gestaltung von Informationssystemen.
BIFOA-Arbeitsbericht Nr. 72/6, Köln 1973

Strunz; Jorgensen (Anwendung graphentheoretischer Verfahren)
Strunz, Horst; Jorgensen, Paul C.:
Anwendung graphentheoretischer Verfahren in der Entscheidungstabellentechnik.
In: Angewandte Informatik, 18. Jg. der "elektronischen datenverarbeitung" 1976, Heft 2, S. 65 - 73

Theory subgroup (DT glossary)
Theory subgroup:
DT glossary preparation-Version 2.
CODASYL DTTG paper. Unveröffentlichtes Manuskript, 23. Januar 1975

Thurner (Entscheidungstabellen)
Thurner, Reinhold:
Entscheidungstabellen. Aufbau-Anwendung-Programmierung.
Düsseldorf 1972

Veinott (Programming decision tables)
Veinott, Cyril G.:
Programming decision tables in FORTRAN, COBOL or ALGOL.
In: Communications of the ACM, Bd. 9, 1966, Nr. 1 S. 31 - 35

Verhelst (Constructing decision tables)
Verhelst, M.:
A technique for constructing decision tables.
In: IAG Quaterly Journal, Bd. 2, 1969, Nr. 1, S. 27 - 36

Verhelst (Conversion of limited-entry decision tables)
Verhelst, M.:
The conversion of limited-entry decision tables to
optimal and near-optimal flowcharts: Two new algorithms.
In: Communications of the ACM, Bd. 15, 1972, Nr. 11,
S. 974 - 980

Vieweg (Konstruktion von Entscheidungstabellen)
Vieweg, Wolfgang:
Die Konstruktion von Entscheidungstabellen.
Wiesbaden 1973

Wedekind (Systemanalyse)
Wedekind, Hartmut:
Systemanalyse. Die Entwicklung von Anwendungssystemen
für Datenverarbeitungsanlagen ..
München 1973

Wirth (Well-structured programs)
Wirth, Niklaus:
On the composition of well-structured programs.
In: ACM Computing surveys, Bd. 6, 1974, Nr. 4, S. 247 - 259

Woods; Hawes (Code generation)
Woods, C. Geoffrey; Hawes, Mary K.:
Optimized code generation from extended-entry decision tables.
In: ACM SIGPLAN notices, Bd. 6, 1971, Nr. 8, S. 74 - 80

Wright (Decision table processors)
Wright, K. R.:
Approaches to decision table processors.
In: Proceedings of the Decision Tables Symposium, Sept. 20-21, New York 1962, S. 41 - 44

Wulf (Case against GOTO)
Wulf, W. A.:
A case against the GOTO.
In: Proceedings of the ACM Annual Conference, Boston, Mass. 1972, S. 791 - 797

Zum Autor:

Prof. Dr. Horst Strunz, geb. 1941, Lehre als Augenoptiker, 1963 Abitur im zweiten Bildungsweg, Studium der Betriebswirtschaftslehre an der Universität zu Köln, 1969 Diplomexamen, 1975 Promotion mit einer grundlegenden Arbeit zur Entscheidungstabellentechnik.

Nach dem Studium 1968 Assistentenstelle am BIFOA Betriebswirtschaftliches Institut für Organisation und Automation an der Universität zu Köln bis 1971, zunächst als Assistent des Studienkreises „Die Wirksamkeit von Programmiersprachen“. Danach Leiter des Projektes „Computergestützte Systemplanung“.

Parallel zur seiner Tätigkeit am BIFOA baute Strunz die Geschäftsstelle Köln des mbp, Mathematischer Beratungs- und Programmierungsdienst GmbH, auf. 1973 erhielt er Prokura, 1981 wurde er zum stellvertretenden und 1983 zum ordentlichen Geschäftsführer ernannt. 1985 gründete er als Hauptgesellschafter die ExperTeam-Gruppe, die er bis 2003 als Geschäftsführer und Vorstand leitete. 2004 bis 2009 war er Geschäftsführer der GeoContent GmbH, seit 2004 führt er die Geschäfte der sbc Strunz Business Consulting GmbH & Co. KG.

1985 Ernennung zum Honorarprofessor an der Universität zu Köln.
1984 bis 1987 war er Vizepräsident der Gesellschaft für Informatik e.V., von 1989 bis 1995 Mitglied der ISDN-Forschungskommission des Landes Nordrhein-Westfalen und 1991 bis 2008 Vorstandsmitglied des FTK Forschungsinstitut für Telekommunikation e.V.

1969 veröffentlichte Strunz seine erste Arbeit über Entscheidungstabellen als BIFOA-Arbeitsbericht. Die außerordentlich positive Reaktion der Praxis veranlasste ihn, ein Seminar über dieses Thema auszuarbeiten. In den folgenden Jahren führte er über 200 Entscheidungstabellen-Seminare durch und sorgte damit für eine rasche Verbreitung dieser Technik. Anfang der 70er-Jahre leitete er bei mbp die Entwicklung des Entscheidungstabellenvorübersetzers VORELLE, der große Verbreitung in der Praxis fand.

1974 bis zur Veröffentlichung des Decision Tables Task Group Reports 1982 war Strunz als Mitglied der Decision Tables Task Group der CODASYL (Conference on Data Systems Languages) in den USA an der Weiterentwicklung der Entscheidungstabellentechnik beteiligt.